
Preface

Three-dimensional surface meshes, composed of collections of planar polygons, are the most common discrete representation of the surface of a virtual shape. These 3D surface meshes need to be inspected in order to understand or evaluate their overall structure or some details. This can be done by extracting relevant geometric or topological features. Such shape characteristics can simplify the way the object is looked at, can help recognition and can describe and categorize it according to specific criteria.

Shape characteristics can be defined in many ways. This book takes the point of view of discrete mathematics, which aims to propose discrete counterparts to concepts mathematically defined in the continuous case. More specifically, in this book, we review how standard geometric and topological notions on surfaces can be defined and computed on a 3D surface mesh, as well as their use for shape analysis. In particular, recent methods are described to extract feature lines having a meaning related to either geometry or topology. Differential estimators such as discrete principal curvatures are detailed as they play a critical role in the computation of salient structures. An emphasis is then made on topology since the global structure and the connectivity of features play an important role in the understanding of a shape. Several applications are finally developed, showing that each of them needs specific adjustments to generic approaches. These applications are related to medicine, geology, botany and other sciences.

Focusing on shape features, the topic of this book is narrower but more detailed than other shape analysis books, which do not, or only briefly, refer to feature definition and computation. It is intended not only to students,

researchers, engineers in computer science and shape analysis, but also to numerical geologists, anthropologists, biologists and other scientists looking for practical solutions to their shape analysis, understanding or recognition problems. We hope that our book will be a useful review of existing work for all of them.

Jean-Luc MARI
Franck HÉTROUY-WHEELER
Gérard SUBSOL
August 2019

Introduction

I.1. Context: 3D shape analysis

Shapes, either from the natural world or manufactured, are more and more digitized for visualization or measurement purposes, among others. This process generally results in 3D surface meshes, which are composed of collections of planar polygons. Such meshes nowadays are the most common discrete representation of the surface of a virtual shape. These meshes are automatically, or sometimes interactively, examined, in order to understand, evaluate or match their overall structure or some details. This process is called *3D shape analysis* and can be done by extracting relevant geometric or topological *features*. Such shape characteristics can simplify the way the object is looked at, can help recognition and can describe and categorize it according to specific criteria. In this book, we will review various mathematical definitions of mesh features and some algorithms to compute them. We will then give a few application examples, where these features are used to globally or locally analyze a 3D shape.

It is important to note that this book is *not* about 3D shape analysis, but only about feature definition and computation. 3D shape analysis has a wider spectrum than feature detection. Among concepts that will not be tackled in this book are global shape descriptor/signature definition or spectral shape analysis. The interested reader can refer to [BIA 14] or [LÉV 10] to learn more about these notions.

Before going into details about surface features from a mathematical and computational point of view, and detailing some applications in Chapters 1–3,

we start by giving some general definitions about meshes, which will be useful in the next chapters. For the interested reader, we also quickly browse how features are defined and calculated in other fields.

1.2. Background on meshes

In many (but not all) contexts where a sampling of a surface is available, this surface is approximated by a mesh. A surface *mesh* is formally defined as a triplet of sets $M = (V, E, F)$, where V is the set of *vertices*, which are points in the 3D Euclidean space \mathbb{R}^3 sampling the surface at stake; E is the set of *edges*, which are segments whose endpoints are vertices of $V : E \subset V \times V$; and F is the set of *faces*, which are polygons whose vertices and edges are vertices and edges of V and E respectively. A volume mesh is defined as a quadruplet of sets (V, E, F, T) . The additional set T is a set of polyhedra whose vertices, edges and faces are part of V , E and F respectively. Since in this book we will focus on the surface of a shape, we will not consider such volume meshes and we will restrict to surface meshes. In practice, several properties are often required. The polygonal faces must be planar. It is often convenient to restrict to triangular faces since such faces are necessarily planar, but sometimes it is useful to allow *quadrangular* faces, i.e. faces with four edges. For most applications, faces must be convex. Here again, restricting to triangles is a way to ensure convexity.

A mesh approximates the surface of an object with a finite number of geometric primitives (points, segments and polygons). The *geometry* of the mesh is thus a simplification of the geometry of the underlying shape and is given by the 3D coordinates of its vertices and by the geometric information carried by the edges and the faces. For example, the normal vectors to the faces are local estimates of the normal vector to the surface. This geometry is decoupled to the *topology* of the mesh, which encodes the neighboring information between primitives. Two meshes can share the same topology but not the same geometry or vice versa (see Figure I.1). Note that what we call here topology is not closely related to the topology of the surface, as will be studied in Chapter 2.

The topology of a mesh is defined with the following notions, which come from the terminology used for graphs in computer science. *Incidence*: two primitives (vertices, edges, faces) are incident to each other if one is a border

of the other. A vertex and an edge are incident to each other if the vertex is an endpoint of this edge. A vertex and a face are incident to each other if the vertex is a vertex of the face. Finally, an edge and a face are incident to each other if the edge is an edge of the face. *Adjacency*: two primitives (vertices, edges, faces) of dimension n are adjacent to each other if they share the same dimension and are incident to the same primitive of dimension $n - 1$ or $n + 1$. Two vertices are adjacent if they are endpoints of the same edge. Two edges are adjacent if they share a common endpoint or a common face. Two faces are adjacent if they share a common edge. Note that in the graph theory, two edges share a common endpoint are said to be incident. We use the term adjacent here for the sake of simplicity. *Neighborhood*: the topological k -neighborhood of a vertex v on a mesh is given by the set of vertices which can be reached from v using at most k adjacency relationships. For example, the 1-neighborhood of v is the set of vertices which share an edge with v , and the 2-neighborhood of v also includes the set of vertices which share an edge with a vertex of the 1-neighborhood of v . Finally, we call *degree* or *valency* of a vertex v the number of 1-neighbors of v . Equivalently, this is the number of edges incident to v .

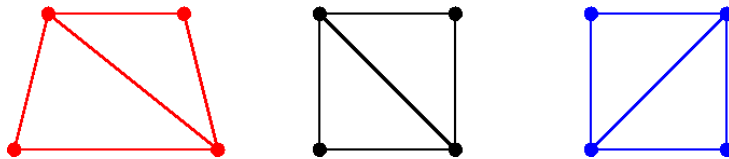


Figure I.1. Three meshes that share either the same topology but not the same geometry (left and middle) or the same geometry but not the same topology (middle and right)

Since the mesh represents the boundary of a volumetric object, it is usually required that locally the surface “looks like” a disk, and that there is no pinch point for example, or three faces sharing an edge. This is formalized through the notion of *manifoldness*, which we will study in Chapter 2. Roughly speaking, a surface is said to be a manifold if the neighborhood of each point can be smoothly deformed to a disk, without shearing or gluing. Alternatively, a surface is a manifold with boundary if the neighborhood of each point can be smoothly deformed to a disk or a half-disk. For example, an egg is a manifold (it is equivalent to a sphere from a topological point of view, although it is not geometrically a sphere), and an egg on which the top has been removed

is a manifold with boundary. More general than a manifold is the notion of *simplicial complex*. A simplicial complex is a mesh M with the following two properties:

1) every edge of a face of F is in E , and every vertex of an edge of E is in V ;

2) the intersection of any two primitives of M is either empty or a primitive of M . For example, the intersection of two different faces is either empty or a vertex or an edge of the mesh. It cannot be a segment or a point inside these faces.

Note that, for example, isolated vertices or faces only connected by a vertex and not an edge are allowed in a simplicial complex while they are not in a manifold mesh.

Meshes are widely used because of several good properties. First, there are relatively easy and fast to create from a set of points. Second, their storage cost is low, since only the 3D coordinates of the vertices and some topological information (e.g. the vertices defining each face) need to be stored. Third, they are adaptable, can handle even complicated geometries and can be used in various scenarios. Finally, they are often easy to process since the neighborhood information is explicitly given by the primitives. However, they face a few drawbacks: the resulting surface is not smooth and they may be difficult to edit since this usually requires the computation of an extended neighborhood for each vertex.

1.3. Definition of a feature

According to the English Oxford Dictionaries [OXF 17], a feature is “*a distinctive attribute or aspect of something*”. This raises two main questions related to the object under study:

- *distinctive*: who is involved in the process of distinguishing?
- *attribute*: how to define an attribute?

Different definitions of features have been proposed for 3D shapes, which differ on their answers to these two questions. For the sake of completeness,

we review here a few of them before introducing the core of our topic, which will be developed in the next chapters.

1.3.1. Surface feature from a topographic point of view

In topography, experts have defined a limited number of Earth (or another planet) surface features. These features are of two kinds: points or lines. The main ones are:

- a *hill* corresponds to a point where the ground slopes down in all directions. In other words, it is a local height maximum;
- a *depression* is a local height minimum;
- a *saddle* is a low point between two hills. The ground slopes up in two opposite directions, but slopes down in the two other, orthogonal directions;
- a *ridge* is a line where at each point the ground slopes up in one direction and down in the three other ones;
- a *valley* is a line where at each point the ground slopes up in three directions and down in the last direction.

These terms can actually be mathematically defined and are not restricted to planets, as we will see in Chapter 1. Other features include draws, spurs, cliffs, cuts and fills [DEP 11]. The main advantage of these definitions is that any non-expert is able to distinguish a feature on a terrain, as soon as he or she is able to check in which directions the ground slopes up or down. The main drawback is that they somehow depend on the choice of the scale: the ground can be slightly sloping down for a few meters in a direction but overall sloping up in the same direction on a longer distance. Hence, more robust and general-purpose definitions have been proposed in other fields.

1.3.2. Surface feature from a perceptual point of view

In visual perception, a feature is usually defined as a salient point or region on a 3D shape, i.e. a point or region that can easily be distinguished from its surroundings by the human eye. Multiple mathematical definitions have been proposed, which usually relate features to surface geometric properties at multiple scales.

In computer graphics, a seminal work has been the introduction of the concept of *mesh saliency* by Lee, Varshney and Jacobs [LEE 05], which extends to 3D meshes the definition proposed by [ITT 98] for 2D images. Earlier works mostly projected a 3D surface into a 2D image to determine its visual features. Lee *et al.* define mesh saliency for a given vertex as a combination of saliencies at different scales. These saliencies are themselves computed as differences between Gaussian-weighted averages of the mean curvature of neighboring vertices computed at a fine and a coarse scale. A feature is then defined as a region on the mesh with high, or locally maximum, saliency values. Note that this original mesh saliency definition is purely geometry-based, but other information such as color can be incorporated.

Almost at the same time Gal and Cohen-Or proposed another purely geometric approach, in which the surface is first approximated by quadric patches [GAL 06]. A descriptor is then defined for each patch, as the center of mass of the patch together with the highest Gaussian curvature across the patch. A salient geometric feature is finally defined as a cluster of descriptors with locally high curvature and a high variance of curvatures. Leifman *et al.* defined *regions of interest* on a 3D mesh according to three properties: vertex distinctness, shape extremities and patch association [LEI 16]. A distinct vertex is defined as a vertex whose spin image descriptor [JOH 99] is far from others, using diffusion distance at three different scales. Contrary to these purely local approaches, Song *et al.* considered global geometric properties of the shape [SON 14]. Specifically, the spectrum of the Laplacian matrix of the mesh is used through a multiscale approach to find the most obvious saliencies.

Following Lee *et al.*, some authors have taken inspiration from visual perception theories to define salient features on a 3D surface. Based on the concept of *visual masking*, Lavoué defined a perceptual measure of noise on a mesh called *roughness*, which is computed at every vertex as an asymmetric difference between the maximum curvature on the mesh and on a smoothed version of it [LAV 09a]. Detected rough vertices are distinct from sharp edge features. Wu *et al.* linearly combine a local property, the visual contrast and a global one, *rarity*, to define saliency [WU 13]. The local contrast is computed by segmenting the mesh into almost planar patches and computing distances between the means of multiscale descriptors of all vertices in neighboring patches. The global rarity value of a vertex is simply the sum of the distances

between the descriptor of this vertex and the descriptors of all other vertices in the mesh. Nader *et al.* proposed a bottom-up approach based on user experiments to estimate the visual contrast on a flat-shaded [NAD 16a] or smooth-shaded [NAD 16b] 3D mesh. Similarly to [LAV 09a], their work is based on the concept of visual masking, as well as others such as *contrast sensitivity* and *visual regularity*.

Since a feature should, by definition, be distinctive, Shilane and Funkhouser introduced the idea of defining a feature on an object with respect to similar objects [SHI 07]. In their approach, a region is said to be distinctive if its shape is consistent with regions on objects of the same class but different from objects in other classes. The shape of a region is described through a multiscale *Harmonic Shape Descriptor* [KAZ 03], which decomposes a spherical region into concentric shells of varying radii. Going further, Chen *et al.* studied the points selected on 3D surfaces by more than 1,000 non-trained users [CHE 12]. The question asked to the users was to select points that the other people were likely to select. They found out that the most obvious feature points can be retrieved at minimum curvature local extrema. Locations for less obvious feature points include symmetry axes and centers of large convex parts. Lau *et al.* extended the concept of mesh saliency to detect the regions on the surface of a 3D object which are the most likely to be grasped, pressed or touched by a human user [LAU 16]. These regions are learned from manually labeled salient regions using deep neural networks.

1.3.3. Surface feature from a machine vision point of view

Automatically computing distinctive features in an image is one of the main goals of computer vision. In this context, features, also called *keypoints*, are usually defined as pixels, or connected sets of pixels, with specific appearance or geometric properties (motion can also be taken into account in the case of videos). For example, a feature can be located at a place of a sharp change in color or texture, or at the edge or the endpoint of an object. Features should represent in a compact manner the relevant information with respect to a specific goal, such as describing the objects in the image. They should also be stable in the image under some geometric transformations, such as rotation or scaling. Feature definition usually comprises two stages: *detection* and *description*. Feature detection aims to compute the relevant keypoints for a given application, while feature description expresses the specific properties

of the image at these points. To each keypoint is associated a neighborhood size called *scale*. This scale is computed in the detection stage and is then used in the description stage. Note that in early detection methods the scale is a user-defined parameter and is the same for all keypoints. Such methods are called *fixed scale* methods. Most of recent works propose *adaptive scale* methods, which automatically compute the scale of each keypoint.

This framework has been extended beyond 2D, in particular to mesh surfaces, by many different approaches. Our goal here is not to survey or compare them, and we refer the reader to recent reviews such as [BRO 12, TOM 13, GUO 14] for these purposes. We only describe here two keypoint detection methods for 3D meshes that have been inspired by image detectors. Other popular feature detectors for 2D images such as SIFT [LOW 04] and SURF [BAY 08] have also been adapted to 3D, but only for range images (i.e. images in which each pixel is associated a depth value, which can be seen as point clouds) in the former case [LO 09] and by voxelizing the mesh in the latter one [KNO 10].

In 2011, Sipiran and Bustos proposed to extend the Harris corner detection method [HAR 88] for 2D images to 3D meshes [SIP 11]. The Harris method is popular because of its robustness to rotations, scaling, noise and changes in illumination. This method considers a neighborhood $W = \{(x_k, y_k)\}$ around each pixel (x, y) and shifts W by a prescribed amount $(\Delta x, \Delta y)$. Then, for each pixel, it computes the sum of squared differences in intensity (the image is converted into a grayscale image) between both neighborhoods. This boils down to the computation of a 2 by 2 matrix $E(x, y)$ involving the partial derivatives of the intensity in x and y , evaluated in the pixels of W . Each pixel is then assigned the value $R(x, y) = \det E(x, y) - \kappa \text{tr}(E(x, y))^2$, with κ being a constant. Corners are then detected as the local maxima of $R(x, y)$. The main difficulty in 3D lies in the computation of the neighborhood. Sipiran and Bustos propose to consider the k -ring neighborhood $N_k(v)$ of the vertex v to be analyzed. The centroid of $N_k(v)$ is calculated, then the vertices in this neighborhood are rotated so that the normal of the best fitting plane going through the centroid aligns with the z -axis. Then, a paraboloid $f(x, y)$ fitting the transformed neighborhood is computed, and the partial derivatives of f in x and y are used to compute the matrix $E(v)$.

Similarly, Zaharescu *et al.* proposed an adapted version of the difference of Gaussians (DoG) framework for meshes, which is called MeshDOG

[ZAH 12]. Originally, this framework convolves the intensity values of the pixels in the image with Gaussian kernels of increasingly large variance, leading to increasingly blurred versions of the image [MAR 80]. The subtraction of two successive results of the convolution enhances the high frequencies in the image and allows the detection of edges, i.e. connected sets of pixels with a sharp intensity change. Alternatively, a *scale-space representation* of the image can be built. This is done by first convolving the image with a Gaussian kernel of a user-defined variance, then iteratively convolving the result with the same Gaussian kernel. The MeshDOG detector proposed by Zaharescu *et al.* builds such a scale-space representation, using geodesic distances between a vertex and its neighbors to define the Gaussian kernel. Feature vertices are then selected among the maxima in this scale-space. Note that the method has been designed independently of the scalar function defined over the mesh. In other words, the value used for each vertex does not necessarily restrict to the local geometry and can include, for example, color information.

1.3.4. Surface feature from a mathematical point of view

We have seen in the previous sections that defining distinctive attributes, to be called features, on a meshed surface can be done in different ways. Features can be defined by experts of the application domain, with the aim of distinguishing relevant locations on the surface. This is, for example, the case in topography (section I.3.1). In visual perception (section I.3.2), features are salient points for the human eye. Their computation often uses properties of the human visual system. Finally, in the machine vision approach (section I.3.3), features are defined with respect to captured geometric or appearance information, so that they possess useful properties such as invariance with respect to some transformations. The process to detect them is usually fully automatic.

In the remainder of this book, we review surface features as defined in two areas of mathematics, namely *geometry* (Chapter 1) and *topology* (Chapter 2). Why these two? Geometry gives tools to express the shape of the surface *locally*, while topology explores it *globally*. Moreover, geometry takes *distances* on the surface into account, while topology seeks for invariants on the surface when it is continuously deformed. Both can be combined and algorithms have been developed to compute topological features with specific

geometric properties, or geometric objects that capture some topology of the surface (see section 2.3).

Computing mathematical features on discrete, point-sampled surfaces such as meshes leads to several problems, since these features are initially defined in the continuous setting. The first of them is the definition of discrete counterparts for such “continuous” features. This can be trivial (e.g. the number of connected components and the genus, section 2.1.1) but is often challenging (e.g. curvature, section 1.3). The second one is the computation itself, which can be prone to inaccuracies (e.g. curvature again) or be time-consuming (e.g. homology groups, section 2.2).

These challenges are discussed in both Chapters 1 and 2, where we first recall how features are defined in the continuous case, before reviewing the definitions and feature computation algorithms in the discrete case. Chapter 3 then details some application fields, such as medicine, geology or botany, for which computing previously defined surface features can be useful.

Geometric Features based on Curvatures

1.1. Introduction

In 1983, the authors of [HAR 83] proposed to use **differential geometry** (i.e. the mathematical analysis of the geometry of 3D curves and surfaces) to describe the shape of a discrete surface (in their case, defined as an elevation image). For this purpose, they introduced the *topographic primal sketch*, which is composed of feature points (as peaks, pits or saddles), feature lines (as ridges or ravines) and feature surface patches (which may be flat). Two years after, the authors of [BRA 85] described another differential geometry-based framework to analyze the shape of 3D surfaces and they applied it on several real examples. They also used feature lines (as curvature lines) and planar surface patches. At the same period, in his dissertation [BES 88a], Besl proposed to compute the differential parameters of a surface in order to create a *HK-sign map* that allows the segmentation of a surface into homogeneous regions where some basic geometric primitives will be fitted.

Since this, extensive research has been done on the extraction and the application of geometric features based on differential geometry. In section 1.2, we propose an overview in a nutshell (but with the mathematical formulas) of differential geometry of surfaces. In section 1.3, we present the main methods that allow us to efficiently compute the differential parameters on a discrete 3D mesh. In the sections 1.4 and 1.5, we focus respectively on line and surface features.

1.2. Some mathematical reminders of differential geometry of surfaces

The following reminders are essentially based on the book [HOS 92]. More details can be found in many mathematics books such as [WEA 55], [CAR 76], [SPI 99], [TOP 05] or [PAT 10].

1.2.1. Fundamental forms and normal curvature

Let Σ a surface of class C^k with $k \geq 3$, which is parameterized by (u, v) . Σ is then defined by the set of points $\mathbf{P}(u, v) = \{x(u, v), y(u, v), z(u, v)\}$.

An infinitesimal displacement around the point \mathbf{P} will be modeled as the vector $d\mathbf{P}$, which will be in the tangent plane defined by the frame $(\mathbf{P}, \frac{\partial \mathbf{P}}{\partial u}, \frac{\partial \mathbf{P}}{\partial v})$:

$$d\mathbf{P} = \frac{\partial \mathbf{P}}{\partial u} du + \frac{\partial \mathbf{P}}{\partial v} dv$$

The norm of this displacement can be computed as:

$$(d\mathbf{P})^2 = \left(\frac{\partial \mathbf{P}}{\partial u}\right)^2 (du)^2 + 2\frac{\partial \mathbf{P}}{\partial u} \frac{\partial \mathbf{P}}{\partial v} dudv + \left(\frac{\partial \mathbf{P}}{\partial v}\right)^2 (dv)^2$$

If we define the terms E , F and G as:

$$E = \left(\frac{\partial \mathbf{P}}{\partial u}\right)^2 \quad F = \frac{\partial \mathbf{P}}{\partial u} \frac{\partial \mathbf{P}}{\partial v} \quad G = \left(\frac{\partial \mathbf{P}}{\partial v}\right)^2$$

we get:

$$(d\mathbf{P})^2 = E(du)^2 + 2F(dudv) + G(dv)^2 \quad [1.1]$$

This expression is called the **first fundamental form** of the surface. Its coefficients E , F and G enable us to calculate $d\mathbf{P}$ that defines the lengths of local curves on the surface around \mathbf{P} . They are also used to define the area of local regions.

Let us now define \mathbf{n} as the normal vector at \mathbf{P} , i.e. the vector going through \mathbf{P} and orthogonal to the tangent plane. For any unit vector \mathbf{t} in the tangent

plane, we can define the plane Π_t that goes through \mathbf{P} and contains \mathbf{n} and \mathbf{t} . Π_t is called a *normal plane* to Σ and cuts Σ along the plane curve S_t , which is called the *normal section* along the direction \mathbf{t} (see Figure 1.1).

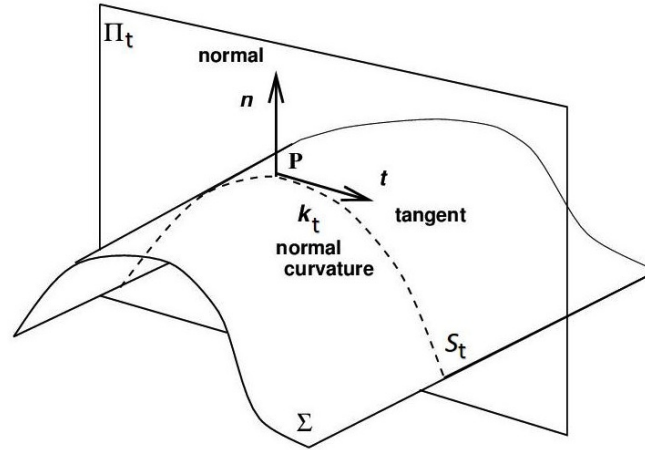


Figure 1.1. At a point \mathbf{P} , the normal plane Π_t is defined by the normal vector \mathbf{n} and the tangent vector \mathbf{t} . Π_t cuts Σ along a plane curve S_t . The curvature of S_t in \mathbf{P} is called the **normal curvature** k_t at \mathbf{P} of the surface Σ along the direction \mathbf{t}

By construction, \mathbf{t} and \mathbf{n} are, respectively, the tangent and the normal vector of S_t at \mathbf{P} . The curvature of the plane curve S_t at \mathbf{P} is an interesting parameter to characterize the local shape of Σ around \mathbf{P} along the direction \mathbf{t} . This curvature, denoted k_t , is given by the first Frenet–Serret formula that relates the tangent and the normal vectors of a planar curve:

$$\frac{d\mathbf{t}}{ds} = k_t \mathbf{n} \quad [1.2]$$

where s is the arclength (also called the curvilinear abscissa) along S_t that is defined by $ds = |d\mathbf{P}|$.

This implies:

$$k_t = \frac{d\mathbf{t}}{ds} \cdot \mathbf{n} \quad [1.3]$$

We can develop the right expression with respect to the parameters (u, v) by using the chain rule:

$$\mathbf{t} = \frac{d\mathbf{P}}{ds} = \frac{\partial\mathbf{P}}{\partial u} \left(\frac{du}{ds} \right) + \frac{\partial\mathbf{P}}{\partial v} \left(\frac{dv}{ds} \right) \quad [1.4]$$

$$\begin{aligned} \frac{d\mathbf{t}}{ds} &= \frac{\partial\mathbf{P}}{\partial u} \left(\frac{d^2u}{ds^2} \right) + \frac{\partial\mathbf{P}}{\partial v} \left(\frac{d^2v}{ds^2} \right) + \frac{\partial^2\mathbf{P}}{\partial u^2} \left(\frac{du}{ds} \right)^2 \\ &\quad + 2 \frac{\partial\mathbf{P}}{\partial u} \frac{\partial\mathbf{P}}{\partial v} \left(\frac{du}{ds} \right) \left(\frac{dv}{ds} \right) + \frac{\partial^2\mathbf{P}}{\partial v^2} \left(\frac{dv}{ds} \right)^2 \end{aligned}$$

A dot product with \mathbf{n} eliminates the two first terms because $\frac{\partial\mathbf{P}}{\partial u}$ and $\frac{\partial\mathbf{P}}{\partial v}$ belong to the tangent plane which is orthogonal to \mathbf{n} . We then get:

$$k_{\mathbf{t}} = \frac{d\mathbf{t}}{ds} \cdot \mathbf{n} = L \left(\frac{du}{ds} \right)^2 + 2M \left(\frac{du}{ds} \right) \left(\frac{dv}{ds} \right) + N \left(\frac{dv}{ds} \right)^2 \quad [1.5]$$

where:

$$L = \mathbf{n} \cdot \frac{\partial^2\mathbf{P}}{\partial u^2} \mathbf{n} = \mathbf{n} \cdot \frac{\partial\mathbf{P}}{\partial u} \cdot \frac{\partial\mathbf{P}}{\partial v} \mathbf{n} = \mathbf{n} \cdot \frac{\partial^2\mathbf{P}}{\partial v^2} \mathbf{n}$$

The parameters L , M and N are called the coefficients of the **second fundamental form** of the surface.

As $ds^2 = (d\mathbf{P})^2$, we can rewrite equation [1.5] by using equation [1.1] as:

$$k_{\mathbf{t}} = \frac{L(du)^2 + 2Mdudv + N(dv)^2}{E(du)^2 + 2Fdudv + G(dv)^2} \quad [1.6]$$

$k_{\mathbf{t}}$ is called the **normal curvature** of Σ at point \mathbf{P} along the direction \mathbf{t} .

Note that another convention exists, where the sign of the curvature is inverted (i.e. $d\mathbf{t}/ds = -k_{\mathbf{t}} \mathbf{n}$). This changes the sign in most of the mathematical expressions given in this chapter, as explained in Table 3.2 of [PAT 10].

1.2.2. Principal curvatures and shape index

Now, if we set $\gamma = dv/du$ which is defined for all direction except along the u isoparametric lines, we can define any tangent vector \mathbf{t} with respect to γ using equation [1.4]:

$$\mathbf{t} = \frac{du}{ds} \left(\frac{\partial \mathbf{P}}{\partial v} + \gamma \frac{\partial \mathbf{P}}{\partial u} \right) \quad [1.7]$$

Let us now parameterize the normal curvature $k_{\mathbf{t}}$ by γ and write it as $k(\gamma)$. As $dv = \gamma du$, equation [1.6] becomes:

$$k(\gamma) = \frac{L + 2M\gamma + N\gamma^2}{E + 2F\gamma + G\gamma^2}$$

$$(L - k(\gamma)E) + 2(M - k(\gamma)F)\gamma + (N - k(\gamma)G)\gamma^2 = 0 \quad [1.8]$$

Let us study the extremal values when the cutting plane Π_t rotates around the axis defined by (\mathbf{P}, \mathbf{n}) . $k(\gamma)$ is then a periodic function as the plane Π_t is invariant after a half-turn. Moreover, as k is a continuous function of γ , there are a maximum and a minimum which are defined by:

$$\frac{dk(\gamma)}{d\gamma} = 0$$

If we differentiate equation [1.8] with respect to γ and if we apply the above extremum condition, we get:

$$(M - k(\gamma)F) + \gamma(N - k(\gamma)G) = 0 \quad [1.9]$$

And if we substitute this result in equation [1.8]:

$$(L - k(\gamma)E) + \gamma(M - k(\gamma)F) = 0 \quad [1.10]$$

We can now write γ as a function of $k(\gamma)$ in the first equation and eliminate γ in the second one, which leads to:

$$(EG - F^2)k(\gamma)^2 + (EN + LG - 2MF)k(\gamma) + LN - M^2 = 0 \quad [1.11]$$

We obtain a quadratic equation in $k(\gamma)$. In the general case, the two roots correspond to the two extrema (one maximum and one minimum) of the normal curvature which are called the **principal curvatures** and are denoted k_1 and k_2 .

The arithmetic mean of the two principal curvatures $k_m = \frac{1}{2}(k_1 + k_2)$ is called the **mean curvature**.

The product of the two principal curvatures $k_g = k_1.k_2$ is called the **Gaussian curvature**. In particular, the Gaussian curvature allows us to define the local shape of surface as:

– $k_g > 0$: the two principal curvatures have the same sign. It implies that all the normal curvatures are of the same sign regardless of the direction. This means that the surface is either locally convex or locally concave;

– $k_g < 0$: the two principal curvatures have an opposite sign. The normal curvature changes its sign and vanishes along a direction. This means that locally the surface goes through its tangent plane.

In the case where there is a unique root, it means that $k_1 = k_2$ and that the normal curvature has a constant value regardless of the direction. This is possible only if the surface is locally plane or spherical around the point **P**. Such a point is then called an **umbilic**.

In order to characterize the shape of a surface with a unique parameter, the **shape index** was proposed in [KOE 92]:

$$s = \frac{2}{\pi} \arctan \frac{k_2 + k_1}{k_2 - k_1} \quad [1.12]$$

s takes its values in $[-1, 1]$. Positive (respectively negative) values correspond to convex (resp. concave) parts and the extremal values 1 or -1 characterize umbilics.

1.2.3. Principal directions and lines of curvature

We call **principal directions** the directions of the tangent vectors along which the normal curvature is extremal. In the general case, we have two

distinct principal directions corresponding to the principal curvatures k_1 and k_2 which are defined by the two tangent vectors \mathbf{t}_1 and \mathbf{t}_2 . Note that we talk of directions so the sense of the tangent vectors must not be taken into account. Thus, the principal direction associated with k_1 can be defined by either \mathbf{t}_1 or $-\mathbf{t}_1$. If \mathbf{P} is an umbilic, we consider that all the directions are principal.

Except in this last case, we can calculate the dot product between the two principal directions. For this, we use the expression of the tangent vector given in equation [1.7]:

$$\mathbf{t}_1 \cdot \mathbf{t}_2 = \frac{du_1}{ds_1} \left(\frac{\partial \mathbf{P}}{\partial u} + \gamma_1 \frac{\partial \mathbf{P}}{\partial v} \right) \cdot \frac{du_2}{ds_2} \left(\frac{\partial \mathbf{P}}{\partial u} + \gamma_2 \frac{\partial \mathbf{P}}{\partial v} \right)$$

By developing and using E , F and G , we get:

$$\mathbf{t}_1 \cdot \mathbf{t}_2 = (E + (\gamma_1 + \gamma_2)F + \gamma_1\gamma_2G) \frac{du_1}{ds_1} \frac{du_2}{ds_2} \quad [1.13]$$

Now, we are going to define a quadratic equation whose solutions are the values corresponding to γ_1 and γ_2 . For this, we write $k(\gamma)$ as a function of γ by using equation [1.9] and eliminate it in equation [1.10]. We obtain:

$$(MG - NF)\gamma^2 + (GL - NE)\gamma + FL - ME = 0 \quad [1.14]$$

This allows us to compute the sum and the product of the roots of this equation as:

$$\begin{aligned} \gamma_1 + \gamma_2 &= \frac{NE - GL}{MG - NF} \\ \gamma_1\gamma_2 &= \frac{FL - ME}{MG - NF} \end{aligned}$$

If we use these two expressions in equation [1.13], we get:

$$\mathbf{t}_1 \cdot \mathbf{t}_2 = 0$$

This proves that the principal directions are **orthogonal** (except when the point is an umbilic).

We can also draw on the surface Σ the curves which are tangent, at each point, to the direction of one of the principal curvatures \mathbf{t}_1 or \mathbf{t}_2 . These curves are called **lines of curvature**. Due to the properties of the principal curvatures, we can conclude that two lines of curvature pass through each non-umbilic point and that they are orthogonal.

As we can see on the example of the ellipsoid in Figure 1.2, the lines of curvature form a network of orthogonal curves all over the surface except at umbilics. Note that for this last particular case, the local pattern of lines of curvature has been intensively studied in [SOT 08].

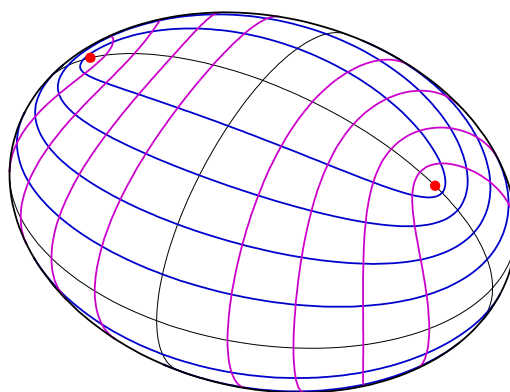


Figure 1.2. Network of lines of curvatures on an ellipsoid: some lines of curvature are visualized in blue and pink, the two visible umbilics (the two others are on the opposite) are localized as red dots¹. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

At any point \mathbf{P} of the surface Σ , we have an orthogonality relationship between the surface normal \mathbf{n} and any tangent vector \mathbf{t} : $\mathbf{n} \cdot \mathbf{t} = 0$.

Now, if we move on the surface along \mathbf{t} by a distance ds , we can then write:

$$\frac{d(\mathbf{n} \cdot \mathbf{t})}{ds} = \frac{d\mathbf{n}}{ds} \cdot \mathbf{t} + \mathbf{n} \cdot \frac{d\mathbf{t}}{ds} = 0$$

¹ <http://commons.wikimedia.org/wiki/File%3Aellipso-KL-NP.svg>. by Ag2gach [CC BY-SA 4.0]

By using equation [1.3], we get:

$$\frac{d\mathbf{n}}{ds} \cdot \mathbf{t} + k_{\mathbf{t}} = 0 \quad \text{or} \quad \frac{d\mathbf{n}}{ds} \cdot \mathbf{t} = -k_{\mathbf{t}} \quad [1.15]$$

If \mathbf{t} is a principal direction, we have seen that $k_{\mathbf{t}}$ is extremal. This implies that the variation of \mathbf{n} according to \mathbf{t} is extremal along the principal directions. Lines of curvature correspond then to the curves of the surfaces where the normal vector of a surface varies the most.

1.2.4. Weingarten equations and shape operator

We can write for any point \mathbf{P} of the surface Σ :

$$\frac{\partial}{\partial u} \left(\mathbf{n} \cdot \frac{\partial \mathbf{P}}{\partial u} \right) = \frac{\partial \mathbf{n}}{\partial u} \cdot \frac{\partial \mathbf{P}}{\partial u} + \mathbf{n} \cdot \frac{\partial^2 \mathbf{P}}{\partial u^2} \quad [1.16]$$

However, as \mathbf{n} is orthogonal to $\frac{\partial \mathbf{P}}{\partial u}$ and $\frac{\partial \mathbf{P}}{\partial v}$, which belong to the tangent plane, we have:

$$\mathbf{n} \cdot \frac{\partial \mathbf{P}}{\partial u} = 0$$

which implies that equation [1.16] is always equal to 0. We have then by using the coefficients of the fundamental forms (see section 1.2.1):

$$\frac{\partial \mathbf{n}}{\partial u} \cdot \frac{\partial \mathbf{P}}{\partial u} = -\mathbf{n} \cdot \frac{\partial^2 \mathbf{P}}{\partial u^2} = -L \quad [1.17]$$

and by exchanging u and v in equation [1.16]:

$$\begin{aligned} \frac{\partial \mathbf{n}}{\partial v} \cdot \frac{\partial \mathbf{P}}{\partial v} &= -\mathbf{n} \cdot \frac{\partial^2 \mathbf{P}}{\partial v^2} = -N \\ \frac{\partial \mathbf{n}}{\partial u} \cdot \frac{\partial \mathbf{P}}{\partial v} &= -\mathbf{n} \cdot \frac{\partial^2 \mathbf{P}}{\partial u \partial v} = -M \\ \frac{\partial \mathbf{n}}{\partial v} \cdot \frac{\partial \mathbf{P}}{\partial u} &= -\mathbf{n} \cdot \frac{\partial^2 \mathbf{P}}{\partial u \partial v} = -M \end{aligned}$$

This set of four equations will allow us to write around \mathbf{P} a linear relationship between the variations of the normal vector and of the point

position. This linear relationship can be represented by a set of two equations called **Weingarten equations**:

$$\begin{aligned}\frac{\partial \mathbf{n}}{\partial u} &= a \frac{\partial \mathbf{P}}{\partial u} + b \frac{\partial \mathbf{P}}{\partial v} \\ \frac{\partial \mathbf{n}}{\partial v} &= c \frac{\partial \mathbf{P}}{\partial u} + d \frac{\partial \mathbf{P}}{\partial v}\end{aligned}$$

or by a linear map called the **shape operator**, which can be represented by the matrix $S(\mathbf{P})$:

$$\begin{pmatrix} \frac{\partial \mathbf{n}}{\partial u} \\ \frac{\partial \mathbf{n}}{\partial v} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \frac{\partial \mathbf{P}}{\partial u} \\ \frac{\partial \mathbf{P}}{\partial v} \end{pmatrix} \quad [1.18]$$

By using equation [1.4], we can then write:

$$\begin{aligned}S(\mathbf{P})\mathbf{t} &= S(\mathbf{P}) \left(\frac{\partial \mathbf{P}}{\partial u} \left(\frac{du}{ds} \right) + \frac{\partial \mathbf{P}}{\partial v} \left(\frac{dv}{ds} \right) \right) \\ &= \frac{\partial \mathbf{n}}{\partial u} \left(\frac{du}{ds} \right) + \frac{\partial \mathbf{n}}{\partial v} \left(\frac{dv}{ds} \right) = \frac{d\mathbf{n}}{ds}\end{aligned}$$

And then, by using equation [1.15], we get the main property of the shape operator:

$$S(\mathbf{P})\mathbf{t} \cdot \mathbf{t} = -k_{\mathbf{t}}$$

for any tangent vector \mathbf{t} :

Now, let us compute the coefficients a , b , c and d . If we take the first Weingarten equation (that corresponds to the first line of the shape operator) and if we apply a dot product with $\frac{\partial \mathbf{P}}{\partial u}$, we get:

$$\frac{\partial \mathbf{n}}{\partial u} \cdot \frac{\partial \mathbf{P}}{\partial u} = a \left(\frac{\partial \mathbf{P}}{\partial u} \right)^2 + b \frac{\partial \mathbf{P}}{\partial v} \cdot \frac{\partial \mathbf{P}}{\partial u}$$

By using equation [1.17] and the coefficients of the first fundamental form (see section 1.2.1), we end up to:

$$-L = aE + bF$$

Similarly, it is easy to show that:

$$-M = aF + bG \quad M = cE + dF \quad N = cF + dG$$

By solving the system of two equations for a and b , we obtain:

$$a = \frac{MF - LG}{EG - F^2} \quad b = \frac{LF - ME}{EG - F^2}$$

And with the system of two equations for c and d , we have:

$$c = \frac{NF - MG}{EG - F^2} \quad d = \frac{MF - NE}{EG - F^2}$$

We can then write explicitly the shape operator matrix:

$$S(\mathbf{P}) = \frac{1}{EG - F^2} \begin{pmatrix} MF - LG & LF - ME \\ NF - MG & MF - NE \end{pmatrix} \quad [1.19]$$

By computing the trace and the determinant, it is easy to see that the eigenvalues of the matrix $S(\mathbf{P})$ are given by equation [1.11]. It means that the eigenvalues correspond to the principal curvatures $k_1(\mathbf{P})$ and $k_2(\mathbf{P})$. This implies that:

$$k_m(\mathbf{P}) = \frac{1}{2} \text{Tr}(S(\mathbf{P})) \quad \text{and} \quad k_g(\mathbf{P}) = \det(S(\mathbf{P}))$$

Now let us find the eigenvectors \mathbf{e}_1 and \mathbf{e}_2 . We can write their coordinates in the tangent frame $(\frac{\partial \mathbf{P}}{\partial u}, \frac{\partial \mathbf{P}}{\partial v})$ by using equation [1.7]:

$$\mathbf{e}_i = \frac{du}{ds} \left(\frac{\partial \mathbf{P}}{\partial u} + \gamma_i \frac{\partial \mathbf{P}}{\partial v} \right)$$

By writing the eigenvalue condition: $S(\mathbf{P})\mathbf{e}_i = \lambda_i \mathbf{e}_i$ where $\lambda_i \in \mathbb{R}$, we get:

$$\begin{cases} a + b\gamma_i = \lambda_i \\ c + d\gamma_i = \lambda_i \gamma_i \end{cases}$$

which gives $b\gamma_i^2 + (a - d)\gamma_i - c = 0$.

If we set $x_i = -\frac{1}{\gamma_i}$, we obtain the equation $-cx_i^2 + (d - a)x_i + b = 0$ which has the same coefficients than equation [1.14]. The two solutions x_i then correspond to the principal direction vectors \mathbf{t}_i whose coordinates are $(1, x_i)$ or $(1, -\frac{1}{\gamma_i})$ in the tangent frame defined below. By construction, the vectors \mathbf{e}_i are orthogonal to \mathbf{t}_i so they also define the principal direction frame. The eigenvectors of the shape operator matrix $S(\mathbf{P})$ are then the principal directions \mathbf{t}_1 and \mathbf{t}_2 and we can write in the principal frame defined by $(\mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$:

$$\frac{\partial \mathbf{n}}{\partial u} = k_1 \frac{\partial \mathbf{P}}{\partial u} \quad \frac{\partial \mathbf{n}}{\partial v} = k_2 \frac{\partial \mathbf{P}}{\partial v} \quad [1.20]$$

1.2.5. Practical computation of differential parameters

It is straightforward to compute differential parameters on the surface of geometric primitives by considering some basic geometric properties.

Thus, on a sphere of radius r , any intersection by a normal plane gives a circle centered at the center of the sphere. The normal curvature along any direction is then equal to $k_t = 1/r$, which means that all the points are umbilics with $k_1 = k_2 = k_m = 1/r$ and $k_g = 1/r^2$.

At any point of a right cylinder of radius r , the surface is locally convex. This means the sign of the normal curvature is positive regardless of the direction. If we intersect the cylinder at a point by a normal plane which goes through the axis, we obtain a straight line. The normal curvature at this point is then equal to 0, and this is an extremal value as all the normal curvatures are positive. This implies that, at any point, we have $k_1 = 0$ and that the corresponding principal direction is along the cylinder generatrix. The other principal direction is orthogonal to the first principal direction, which implies that it defines a transverse plane, cutting the cylinder as a circle of radius r . We deduce then that $k_2 = 1/r$.

For more complex geometric surfaces given by a parametric representation (u, v) , we usually compute the parameters of the first and second fundamental forms. This then allows us to solve equation [1.11] to compute the principal curvatures k_1 and k_2 (only one value in the case of an umbilic). Then, we can solve equation [1.14] to get two values (except for umbilics) of γ in order to compute the two principal directions \mathbf{t}_1 and \mathbf{t}_2 by equation [1.7].

We can also compute the coefficients of the shape operator matrix by using formulas given in equation [1.19]. The eigenvalues and the eigenvectors of this 2×2 matrix give the principal curvatures and directions.

Note that in the specific case, called *Monge form*, where the surface Σ is defined by $\mathbf{P}(u, v) = (u, v, w(u, v))$, the equations become simpler (see, for example, [HAM 93]) as we have:

$$E = 1 + \left(\frac{\partial w}{\partial u}\right)^2 \quad F = \frac{\partial w}{\partial u} \frac{\partial w}{\partial v} \quad G = 1 + \left(\frac{\partial w}{\partial v}\right)^2 \quad [1.21]$$

With $D = 1 + \left(\frac{\partial w}{\partial u}\right)^2 + \left(\frac{\partial w}{\partial v}\right)^2$ we get:

$$L = \frac{1}{\sqrt{D}} \frac{\partial^2 w}{\partial u^2} \quad M = \frac{1}{\sqrt{D}} \frac{\partial^2 w}{\partial u \partial v} \quad N = \frac{1}{\sqrt{D}} \frac{\partial^2 w}{\partial v^2}$$

$$k_g(\mathbf{P}) = \frac{1}{D^2} \left(\frac{\partial^2 w}{\partial u^2} \frac{\partial^2 w}{\partial v^2} - \left(\frac{\partial^2 w}{\partial u \partial v} \right)^2 \right)$$

$$k_m(\mathbf{P}) = \frac{1}{2D^{\frac{3}{2}}} \left(\left(1 + \frac{\partial w^2}{\partial v} \right) \frac{\partial^2 w}{\partial u^2} - 2 \frac{\partial w}{\partial u} \frac{\partial w}{\partial v} \frac{\partial^2 w}{\partial u \partial v} + \left(1 + \frac{\partial w^2}{\partial u} \right) \frac{\partial^2 w}{\partial v^2} \right)$$

which allows us to compute easily the principal curvatures $k_1(\mathbf{P})$ and $k_2(\mathbf{P})$.

We can also obtain mathematical formulations when we use other type of parameterization as an implicit equation involving the three coordinates (see, for example, [GOL 05]).

1.2.6. Euler's theorem

Let us parameterize the surface Σ by the lines of curvature. In this case, at a point \mathbf{P} , the tangent vectors of the isoparametric curves correspond to the principal directions $(\mathbf{t}_1, \mathbf{t}_2)$ which are orthogonal. We have then:

$$F = \frac{\partial \mathbf{P}}{\partial u} \frac{\partial \mathbf{P}}{\partial v} = 0$$

and by writing equation [1.9] with $\gamma = 0$, we get $M = 0$. Equation 1.3 which gives the curvature along any tangent direction \mathbf{t} then becomes:

$$k_{\mathbf{t}} = \frac{L(du)^2 + N(dv)^2}{E(du)^2 + G(dv)^2} \quad [1.22]$$

In particular, the principal curvatures that correspond respectively to $dv = 0$ and $du = 0$ are given by:

$$k_1 = L/E \quad k_2 = N/G \quad [1.23]$$

Any tangent vector $\mathbf{t}(\theta)$ which forms an angle θ with the principal direction \mathbf{t}_1 can be written as $\mathbf{t}(\theta) = \cos \theta \mathbf{t}_1 + \sin \theta \mathbf{t}_2$ and we can write according to equation [1.4]:

$$\mathbf{t}(\theta) \cdot \mathbf{t}_1 = \cos \theta = \left(\frac{\partial \mathbf{P}}{\partial u} \left(\frac{du}{ds} \right) + \frac{\partial \mathbf{P}}{\partial v} \left(\frac{dv}{ds} \right) \right) \cdot \mathbf{t}_1$$

As \mathbf{t}_1 is a unit vector, we have:

$$\frac{\partial \mathbf{P}}{\partial u} = \left\| \frac{\partial \mathbf{P}}{\partial u} \right\| \mathbf{t}_1$$

As $\frac{\partial \mathbf{P}}{\partial v}$ is orthogonal to \mathbf{t}_1 , this gives:

$$\mathbf{t}(\theta) \cdot \mathbf{t}_1 = \left\| \frac{\partial \mathbf{P}}{\partial u} \right\| \frac{du}{ds} = \sqrt{E} \frac{du}{ds} = \cos \theta$$

Identically, we get:

$$\sqrt{G} \frac{dv}{ds} = \sin \theta$$

Based on the above results, we can now write:

$$k_1 \cos^2 \theta + k_2 \sin^2 \theta = L \left(\frac{du}{ds} \right)^2 + N \left(\frac{dv}{ds} \right)^2$$

As $F = 0$, we have by equation [1.1]: $ds^2 = (d\mathbf{P})^2 = E(du)^2 + G(dv)^2$. We can then replace the denominator and we obtain by equation [1.22]:

$$k_1 \cos^2 \theta + k_2 \sin^2 \theta = \frac{L(du)^2 + N(dv)^2}{E(du)^2 + G(dv)^2} = k_{\mathbf{t}}(\theta) \quad [1.24]$$

This formula, known also as **Euler's theorem**, allows us to compute the value of the directional curvature at a point, along any tangent vector, given only the principal curvatures and directions.

1.2.7. Meusnier's theorem

Let us study the curve \mathcal{S}_t around \mathbf{P} . Locally, we can represent this plane curve in the frame $(\mathbf{P}, \mathbf{t}, \mathbf{n})$ by a Cartesian representation $y = f(x)$. We can write Taylor's expansion of f around \mathbf{P} :

$$f(x) = f(0) + f'(0)x + \frac{1}{2}f''(0)x^2 + \epsilon$$

At \mathbf{P} , $f(0) = 0$ and $f'(0) = 0$ as \mathbf{t} is a tangent vector to \mathcal{S}_t . We can then write:

$$f''(0) = \lim_{\mathbf{P}} \frac{2f(x)}{x^2}$$

The general formula for the curvature k of a 2D function f is given by:

$$k = \frac{f''(x)}{(1 + f'^2(x))^{\frac{3}{2}}}$$

If we apply it to the curve \mathcal{S}_t at \mathbf{P} , we get:

$$k_{\mathcal{S}_t}(\mathbf{P}) = f''(0) = \lim_{\mathbf{P}} \frac{2f(x)}{x^2} \quad [1.25]$$

Now, let us consider, at point \mathbf{P} , a normal plane Π_t directed by the tangent vector \mathbf{t} . If we rotate this plane along the axis (\mathbf{P}, \mathbf{t}) by an angle θ , we get a new plane that we call Π_t^θ . This plane cuts the surface Σ along the curve \mathcal{S}_t^θ .

The normal vector of \mathcal{S}_t^θ at \mathbf{P} is \mathbf{n}^θ whereas the tangent vector is, by construction, \mathbf{t} .

The plane curve \mathcal{S}_t^θ can be parameterized by $(x(t), y(t))$ in the frame of the plane Π_t^θ defined by $(\mathbf{t}, \mathbf{n}^\theta)$. We can compute its 2D curvature at \mathbf{P} called k_t^θ using equation [1.25]:

$$k_t^\theta = \lim_{\mathbf{P}} \frac{2y(t)}{x^2(t)}$$

However, we can also parameterize \mathcal{S}_t^θ as a 3D curve $(X(u), Y(u), Z(u))$ in the frame $(\mathbf{t}, \mathbf{n}, \mathbf{t} \times \mathbf{n})$. It is then easy to see that $X(t) = x(t)$ and $Y(t) = y(t)/\cos\theta$.

The previous equation then becomes:

$$k_t^\theta = \frac{1}{\cos\theta} \lim_{\mathbf{P}} \frac{2Y(t)}{X^2(t)}$$

But we have seen in equation [1.25] that such a limit expression corresponds to the curvature at \mathbf{P} of a curve tangent to \mathbf{t} , traced in the plane Π_t . In other words, this is the normal curvature k_t and we then get:

$$k_t^\theta = \frac{k_t}{\cos\theta} \quad [1.26]$$

This equation is called **Meusnier's theorem**. It is useful to compute the curvature of any inclined section of the surface Σ with respect to the curvature of the normal section with the same tangent vector.

1.2.8. Local approximation of the surface

In a general way, we can approximate the shape of the surface Σ around the point \mathbf{P} by the second order expansion:

$$d\mathbf{P} = \frac{\partial\mathbf{P}}{\partial u} du + \frac{\partial\mathbf{P}}{\partial v} dv + \frac{1}{2} \left(\frac{\partial^2\mathbf{P}}{\partial u^2} du^2 + 2 \frac{\partial\mathbf{P}}{\partial u} \frac{\partial\mathbf{P}}{\partial v} dudv + \frac{\partial^2\mathbf{P}}{\partial v^2} dv^2 \right)$$

If we perform a projection on the normal vector \mathbf{n} , we eliminate the tangent vectors $\frac{\partial \mathbf{P}}{\partial u}$ and $\frac{\partial \mathbf{P}}{\partial v}$. We then get an equation of the shape relatively to z , which is the height with respect to the tangential plane at \mathbf{P} . With the coefficients of the second fundamental form, we can write it as:

$$z(u, v) = d\mathbf{P} \cdot \mathbf{n} = \frac{1}{2}(Ldu^2 + 2Mdudv + Ndv^2)$$

Now, let us assume that the surface Σ is parameterized by the lines of curvature. For each point \mathbf{P} , we have the local frame $(\mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$. If we move \mathbf{P} along the isoparametric line $dv = 0$, we can define the x coordinate as:

$$x = d\mathbf{P} \cdot \mathbf{t}_1 = \frac{\partial \mathbf{P}}{\partial u} du \cdot \mathbf{t}_1 = \sqrt{E} du$$

Similarly, we can define $y = \sqrt{G} dv$.

We have seen in section 1.2.6 that $M = 0$ in the case of this specific parameterization which gives:

$$z(x, y) = \frac{1}{2} \left(L \frac{x^2}{E} + N \frac{y^2}{G} \right)$$

and from equations [1.23], we infer:

$$z(x, y) = \frac{1}{2}(k_1 x^2 + k_2 y^2)$$

Such equation allows us to represent the local shape of the surface at point \mathbf{P} at order 2. In particular, if k_1 and k_2 share the same sign (which is equivalent to $k_g > 0$), the shape is locally an ellipsoid and the point is said to be *elliptical*. If the signs are opposite (i.e. $k_g < 0$), the point is said to be *hyperbolic* and the shape looks locally like as a saddle. If one principal curvature is equal to zero (i.e. $k_g = 0$), the point is said to be *parabolic*.

1.2.9. Focal surfaces

For any point \mathbf{P} of Σ where the principal curvatures are not equal to 0, we can define the **centers of principal curvatures** $\mathbf{C}_1(\mathbf{P})$ and $\mathbf{C}_2(\mathbf{P})$ as:

$$\mathbf{C}_1(\mathbf{P}) = \mathbf{P} - \frac{1}{k_1(\mathbf{P})} \mathbf{n}(\mathbf{P}) \quad \mathbf{C}_2(\mathbf{P}) = \mathbf{P} - \frac{1}{k_2(\mathbf{P})} \mathbf{n}(\mathbf{P})$$

By continuity, the set of $C_1(\mathbf{P})$ (resp. $C_2(\mathbf{P})$) defines a surface E_1 (resp. E_2) which is called the **focal surface** (or **surface of centers** or **evolute surface**) of the principal curvature k_1 (resp. k_2).

There is a specific relationship between Σ and its focal surfaces E_1 and E_2 . The following demonstration is adapted from [YU 07] (see also section 10.4 of [POR 01]).

We can estimate the variation of a point \mathbf{C}_1 over E_1 when \mathbf{P} moves. For this, we use the principal frame $(\mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$ and we can write:

$$\frac{\partial \mathbf{C}_1}{\partial u} = \frac{\partial \mathbf{P}}{\partial u} - \frac{1}{k_1} \frac{\partial \mathbf{n}}{\partial u} + \frac{1}{k_1^2} \frac{\partial k_1}{\partial u} \mathbf{n}$$

We have seen in equation [1.20] that $\frac{\partial \mathbf{n}}{\partial u} = k_1 \frac{\partial \mathbf{P}}{\partial u}$. It results:

$$\frac{\partial \mathbf{C}_1}{\partial u} = \frac{1}{k_1^2} \frac{\partial k_1}{\partial u} \mathbf{n}$$

$\frac{\partial \mathbf{C}_1}{\partial u}$ defines a tangent vector on E_1 . Thus, the above equation shows that the normal vector \mathbf{n} is tangent to E_1 .

Now, we can estimate the variation of a point \mathbf{C}_1 over E_1 when \mathbf{P} moves along the line of curvature defined by the principal curvature \mathbf{t}_2 :

$$\frac{\partial \mathbf{C}_1}{\partial v} = \frac{\partial \mathbf{P}}{\partial v} - \frac{1}{k_1} \frac{\partial \mathbf{n}}{\partial v} + \frac{1}{k_1^2} \frac{\partial k_1}{\partial v} \mathbf{n}$$

We have seen in equation [1.20] that $\frac{\partial \mathbf{n}}{\partial v} = k_2 \frac{\partial \mathbf{P}}{\partial v}$. It results:

$$\frac{\partial \mathbf{C}_1}{\partial v} = \left(1 - \frac{k_2}{k_1}\right) \frac{\partial \mathbf{P}}{\partial v} + \frac{1}{k_1^2} \frac{\partial k_1}{\partial v} \mathbf{n}$$

$\frac{\partial \mathbf{C}_1}{\partial v}$ defines a tangent vector on E_1 . As we have seen that \mathbf{n} is also tangent to E_1 , it results from the above equation that \mathbf{t}_2 is tangent to E_1 . As $(\mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$ forms an orthonormal frame, we can deduce that \mathbf{t}_1 is normal to E_1 .

Similarly, we can show that \mathbf{n} is tangent to E_2 and that \mathbf{t}_2 is normal to E_2 . This shows that E_1 and E_2 can be defined as the envelopes of the normal vectors to Σ and that the normal vectors of E_1 and E_2 correspond to the principal directions of Σ .

Note that in [HAG 92], the authors propose a generalization of the definition of the focal surfaces. Points of the “generalized focal surface” have the following form:

$$\mathbf{F}(\mathbf{P}) = \mathbf{P} + \alpha f(k_1(\mathbf{P}), k_2(\mathbf{P}))\mathbf{n}(\mathbf{P})$$

with α a real value called the scale factor. Different functions f are proposed (as $f = k_1 k_2 = k_g$ or $f = k_1^2 + k_2^2$). By displaying this generalized focal surface, it is possible to emphasize some characteristics of the surface shape (see section 5.4 of [HAH 08]).

1.3. Computation of differential parameters on a discrete 3D mesh

1.3.1. Introduction

We have seen in the previous section the formulas to compute differential parameters on a continuous surface. In the case of a discrete 3D mesh, the problem becomes much more complex as we lose the analytic definition of the surface. We then need to either introduce an approximation scheme based on sparse measurements, or make some hypotheses in order to fit a continuous representation.

Many methods have been proposed to compute differential parameters on a 3D mesh and we can find quite complete surveys in [MAG 07] or [GAT 06]. In the following sections, for each type of feature, we list the main categories of methods and present with more details one algorithm (very often, one of the first published). Note that sometimes, the original implementation will be simplified for a better comprehension.

1.3.2. Some notations

We will assume that the 3D mesh \mathcal{M} is given by its set of *vertices* $\mathbf{P}_i(x_i, y_i, z_i)$, the set of *edges* e_j linking two vertices and the set of *facets* F_k

composed of edges (3 in the standard case of triangular facets). We will denote the oriented edge linking \mathbf{P}_i and \mathbf{P}_j as $\mathbf{e}_i^j = \mathbf{P}_i\mathbf{P}_j$.

For each vertex \mathbf{P}_i , we can define its *neighborhood* $N(\mathbf{P}_i)$ that contains all the vertices which are connected to \mathbf{P}_i . This direct neighborhood is also called the *1-ring* of the vertex \mathbf{P}_i . We can then compute the neighborhood of all the vertices belonging to $N(\mathbf{P}_i)$ and fuse them which result in a more general neighborhood called the *2-ring* of the vertex \mathbf{P}_i . Recursively, we can define the *n-ring* of the vertex \mathbf{P}_i .

We assume that the 3D mesh \mathcal{M} is orientable, which means that the ordering of vertices in a facet is consistent between two adjacent facets: when two vertices of a common edge are ordered in one direction in a facet, they must be ordered in the opposite direction in the other facet. This allows us to define consistently the two sides of \mathcal{M} .

At each vertex \mathbf{P}_i , we can define the *normal vector* \mathbf{n}_i to the mesh \mathcal{M} . This is the vector which is locally “orthogonal” to the surface. Note that on a discrete mesh, this notion of orthogonality is not straightforward to define at a vertex where many planar faces intersect. When normal vectors are considered on a closed 3D mesh, they are generally defined outward-pointing (i.e. pointing towards the exterior of the mesh).

1.3.3. Computing normal vectors

To compute differential parameters, most of the methods require first the computation of the discrete normal vector \mathbf{n}_i at each vertex \mathbf{P}_i of a 3D mesh. This is a critical step as it influences a great deal the following of the process.

Many algorithms have been proposed and we can find a review of them in [JIN 05]. In the following, we focus on an algorithm where the normal vector is computed by averaging the normal vectors to the adjacent facets weighted by their areas. This is a particular case of the class of “mean weighted methods” (see, in particular, [MAX 99]), which are efficient and easy to implement. Note that we assume that all the triangles of the mesh are oriented consistently.

It is generally accepted that the computed normal vector should point “outside” the mesh. This depends, in fact, on the initial choice of facet orientation, which will define the direction of the cross-product in the

algorithm. In most cases, the process of acquisition and design of the 3D mesh results in a correct orientation over all its facets. Nevertheless, some methods have been proposed (see, for example, [TAK 14]) to correct inconsistencies in facet orientation or in normal direction.

Input: Vertex \mathbf{P}_i
Output: Normal vector \mathbf{n}_i
begin
 Take all the neighbor vertex $\mathbf{P}_j \in N(\mathbf{P}_i)$.
 All these vertices form a set of facets F_i^j which shares the vertex \mathbf{P}_i .
forall facet F_i^j (which can be triangular or not) **do**
 Take the vectors \mathbf{e} and \mathbf{e}' corresponding to the two edges of the facet F_i^j which are incident to \mathbf{P}_i
 Define the triangle $(\mathbf{P}_i, \mathbf{e}, \mathbf{e}')$ and compute its area A_i^j and its normal vector \mathbf{n}_i^j by using the cross product: $A_i^j = \frac{1}{2} \|\mathbf{e} \times \mathbf{e}'\|$ and $\mathbf{n}_i^j = (\mathbf{e} \times \mathbf{e}') / 2A_i^j$
end
 Compute the normal vector at \mathbf{P}_i by averaging all the normal vectors of the neighborhood weighted by their areas:

$$\mathbf{n}_i = \frac{1}{\sum_j A_i^j} \sum_j A_i^j \mathbf{n}_i^j$$

end

Algorithm 1: Computing the normal vector \mathbf{n}_i at vertex \mathbf{P}_i .

Note that the problem is much complex if we have an unstructured 3D point cloud instead of a 3D mesh. As we have no connection information anymore, we define the neighborhood of \mathbf{P}_i as the set of points $N_\sigma(\mathbf{P}_i)$, which are at a distance less than a given threshold σ . We can estimate the plane which fits the best the set of points $N_\sigma(\mathbf{P}_i)$ by the least-square criterion. The normal vector at point \mathbf{P}_i is then given by a normalized vector orthogonal to this plane. Note that this normal vector is non-oriented and that some post-processing is required to associate a consistent direction (see the above paragraph). A major concern is that σ has a large influence on the result quality and we can find in [MIT 04] a method to find the optimal neighborhood size using local information.

Input: Vertex \mathbf{P}_i , normal \mathbf{n}_i

Output: Principal curvatures $k_1(\mathbf{P}_i)$ and $k_2(\mathbf{P}_i)$ and the associated principal directions $\mathbf{t}_1(\mathbf{P}_i)$ and $\mathbf{t}_2(\mathbf{P}_i)$

begin

Define the plane Π passing through \mathbf{P}_i and orthogonal to \mathbf{n}_i . This corresponds to the tangent plane to the 3D mesh at \mathbf{P}_i .

Build an orthogonal coordinate system centered in \mathbf{P}_i composed of the normal \mathbf{n}_i and two orthogonal vectors \mathbf{i}' and \mathbf{j}' taken at random in Π . For example, we can take in the canonical frame $(\mathbf{i}, \mathbf{j}, \mathbf{k})$, after normalization, $\mathbf{i}' = \mathbf{n}_i \times \mathbf{i}$ and $\mathbf{t}_2 = \mathbf{n}_i \times \mathbf{t}_1$.

forall neighbor vertex $\mathbf{P}_j \in N(\mathbf{P}_i)$ **do**

 Compute the coordinates (u_j, v_j, w_j) of \mathbf{P}_j in this system. w_j corresponds then to the distance between \mathbf{P}_j and the plane Π .

end

Approximate locally the surface of the 3D mesh by using a bivariate polynomial of order 2 (which corresponds also to the Taylor expansion of order 2 around \mathbf{P}_i):

$$w(u, v) = au^2 + 2buv + cv^2$$

We want that $w(0, 0) = 0$ as the origin of the coordinate system is \mathbf{P}_i and that $w(u_j, v_j)$ is as close as possible to w_j . If we introduce the squared distance:

$$E = \sum_{N(\mathbf{P}_i)} [(au_j^2 + 2bu_jv_j + cv_j^2) - w_j]^2$$

the objective is to minimize E in order to find the values a , b and c . This can be done by least-square minimization (see for example section 15.1 of [PRE 07]).

We have now a surface around \mathbf{P}_i given by $\mathbf{P}(u, v) = (u, v, w(u, v))$ and by using equations [1.21] and [1.11], we can easily compute the principal curvatures $k_1(\mathbf{P}_i)$ and $k_2(\mathbf{P}_i)$ and the principal directions $\mathbf{t}_1(\mathbf{P}_i)$ and $\mathbf{t}_2(\mathbf{P}_i)$.

end

Algorithm 2: Computing differential parameters by fitting locally a continuous parametric function.

1.3.4. *Locally fitting a parametric surface*

In order to compute the differential parameters, a first idea is to fit locally a continuous parametric surface $f(u, v)$ on the 3D mesh around P_i , which allows us to use equations of section 1.2.5. One of the first algorithms was proposed in [HAM 93] (in fact, we can find a similar method in an older paper [STO 92] but it was applied to depth images). The algorithm aims at fitting a quadratic surface around each vertex in order to get a formal computation of the differential parameters:

In [PET 02], it is emphasized that this class of methods relies heavily on the accuracy of the surface normal \mathbf{n}_i . The authors propose then a more sophisticated method based on an extended quadric surface, which allows us to refine iteratively the computation of the normal vector.

In [CAZ 05a], the authors propose to use polynomial fitting with an order higher than 2. By increasing the order, they can compute derivatives of the differential parameters which can be used to define some feature lines. This method has been made available in the open-source C++ Computational Geometry Algorithms Library CGAL² (see also [CAZ 08b]).

1.3.5. *Discrete differential geometry operators*

The objective of discrete differential geometry (DDG) is to define geometry differential operators directly from measurements performed on the discrete elements of the 3D mesh (as the area of a facet or the angle between two vertices), in other words, without any approximation by a continuous function (see [CRA 13] for more details).

In the following, we illustrate the classical formulas of DDG by presenting some approximations. The idea is to model the local shape of the 3D mesh (which is assumed to be triangulated) at vertex P_i by a simple geometric primitive, which allows the computation of either the Gaussian (based on a sphere [MES 07]) or the mean curvature (based on cylinders [MES 12, DYN 01]).

² http://doc.cgal.org/latest/Jet_fitting3/classCGAL_1_1Monge_via_jet_fitting.html.

Gaussian curvature

Let us define a cone Γ defined by its apex \mathbf{P} , its angle α and its slant height l (see Figure 1.3, left). The radius of the cone base is then $R = l \sin \alpha$ which gives a perimeter of $2\pi l \sin \alpha$. If we cut Γ along a generatrix, unfold the cone surface and develop it on a plane, we have a sector of disk of radius l and of angle θ (see Figure 1.3, right). Note that $2\pi - \theta$ is called the *deficit angle* or the *defect angle* at \mathbf{P} .

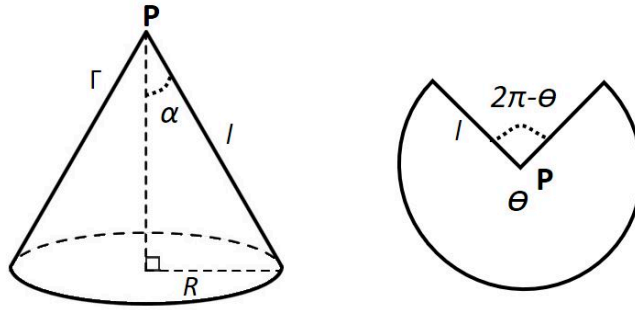


Figure 1.3. Left: the cone Γ is defined by its apex \mathbf{P} , its angle α and its slant height l . Right: the cone is now developed on a plane and becomes a sector of disk of angle θ

The perimeter of the disk sector is θl and by construction, it is equal to the perimeter of the cone base which leads to $2\pi l \sin \alpha = \theta l$ or $\sin \alpha = \frac{\theta}{2\pi}$.

Now let us introduce a sphere of radius r into the cone, it stabilizes along a tangent circle of radius R . Let A the area of the spherical cap Σ which is above this tangent circle. If h is the height of the spherical cap (see Figure 1.4, left), we get $A = 2\pi r h$ or $h = A/2\pi r$. By using the relationships in a squared triangle, we get $h = r(1 - \sin \alpha) = r(1 - \theta/2\pi)$. Based on the two previous expressions, we infer that $1/r^2 = (1/A)(2\pi - \theta)$.

If we approximate the Gaussian curvature at \mathbf{P} by the Gaussian curvature of the sphere which is equal to $1/r^2$, we get:

$$k_g(\mathbf{P}) = \frac{1}{A}(2\pi - \theta)$$

In a discrete 3D mesh, the set of facets F_i^j which share the vertex \mathbf{P}_i forms a generalized cone (with a regular shape if the mesh is locally convex or concave). By using the Gauss–Bonnet theorem, we can demonstrate (see, for example, [MEY 03], [ALB 05] or [MEE 00]) a similar formula:

$$k_g(\mathbf{P}_i) = \frac{1}{A_i} (2\pi - \sum_j \theta_i^j)$$

where θ_i^j is the angle of the facet F_i^j at the vertex \mathbf{P}_i and A_i is an “area” defined around the vertex \mathbf{P}_i (see Figure 1.4, right).

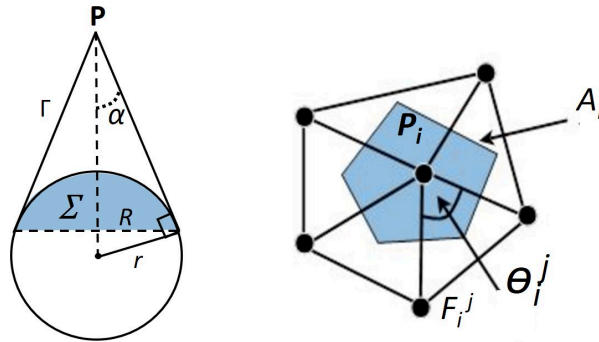


Figure 1.4. Left: if we introduce a sphere of radius r into the cone, it stabilizes along a tangent circle of radius R ; the spherical cap Σ of height h and area A will be above this circle. Right: the discrete Gaussian curvature at vertex \mathbf{P}_i is based on the angles of the adjacent facets F_i^j at \mathbf{P}_i which defines the discrete “deficit angle” and on the “area” defined around \mathbf{P}_i

As emphasized in [DYN 01], there are, in fact, different ways of defining the area A_i , which result in different curvature values. The most commonly used area definitions are:

– *barycentric area* which considers that a facet is equally shared by three vertices. We then get $A_i = \frac{1}{3} \sum_j A_i^j$ where A_i^j is the area of facet F_i^j ;

– *mixed area* (sometimes also called *Voronoi area*) defined in [MEY 03]. In this case, for each facet F_i^j , we perform a Voronoi tessellation based on the three vertices and we take into account only the area of the Voronoi cell including \mathbf{P}_i .

We can find a discussion about these two definitions in [XU 06].

Mean curvature

Suppose that we replace each edge e_i^j between P_i and P_j by a small cylinder portion Σ_i^j of radius r_i^j that joins the adjacent faces tangentially (see Figure 1.5, left). We have seen in section 1.2.5 that for a cylinder of radius r , we have one principal curvature (along a generatrix) equal to 0 and the other (along a circle) equal to $1/r$. So for each point of Σ_i^j , we have $k_1 = 1/r_i^j$, $k_2 = 0$ and the mean curvature is $k_m^j = 1/2r_i^j$.

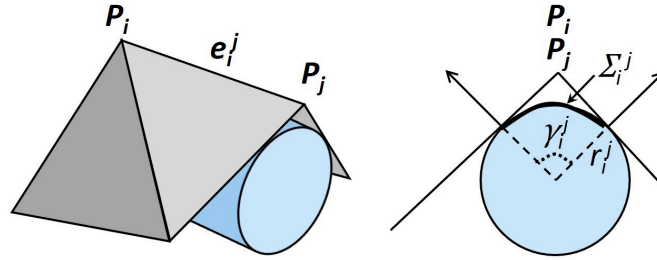


Figure 1.5. The edge e_i^j between the vertex P_i and a neighbor vertex P_j is approximated by a tangent cylinder portion Σ_i^j of radius r_i^j

Let us define γ_i^j the angle between the normal vectors of the two adjacent facets (see Figure 1.5, right). The area of the cylinder portion Σ_i^j is equal to $A_i^j = \gamma_i^j r_i^j \|\mathbf{e}_i^j\|$. This leads to $k_1 = \gamma_i^j \|\mathbf{e}_i^j\| / A_i^j$ and $k_m^j = \gamma_i^j \|\mathbf{e}_i^j\| / 2A_i^j$.

Now, the idea is to approximate the mean curvature of P_i by averaging the mean curvature of each Σ_i^j weighted by their areas. As each area concerns two facets, we have to divide the result by 2. We then get:

$$k_m(\mathbf{P}_i) = \frac{1}{A_i} \sum_j \frac{1}{2} A_i^j k_m^j = \frac{1}{A_i} \sum_j \frac{1}{2} A_i^j \frac{\gamma_i^j \|\mathbf{e}_i^j\|}{2A_i^j} = \frac{1}{4A_i} \sum_j \gamma_i^j \|\mathbf{e}_i^j\|$$

where A_i is the area around P_i . As for the Gaussian curvature, it is proposed to take for A the aforementioned barycentric area or mixed area.

Another discrete method to compute the mean curvature was proposed in [MEY 03]. It is based on the discrete Laplace–Beltrami operator applied to the

mean curvature flow and results also in a weighted sum of the length of the incident edges at \mathbf{P}_i , but with different angles:

$$k_m(\mathbf{P}_i) = \frac{1}{2A} \sum_j (\cot \alpha_i^j + \cot \beta_i^j) \|\mathbf{e}_i^j\|$$

where α_i^j and β_i^j are the measurements of the angles opposite to the edge \mathbf{e}_i^j and A is the *mixed area*. A discussion between the two formulas can be found in [MES 12, SIM 13].

In its simplest form, the algorithm to compute Discrete Differential Geometry operators is then:

Input: Vertex \mathbf{P}_i

Output: Gaussian and mean curvatures k_g and k_m , principal curvatures $k_1(\mathbf{P}_i)$ and $k_2(\mathbf{P}_i)$ with the associated principal directions $\mathbf{t}_1(\mathbf{P}_i)$ and $\mathbf{t}_2(\mathbf{P}_i)$

begin

forall facet F_i^j (which are assumed triangular) of \mathcal{F}_i **do**

 | Compute the area A_i^j .

 | Compute the angle θ_i^j at \mathbf{P}_i .

end

forall edge \mathbf{e}_i^j **do**

 | Compute the angle γ_i^j between the two adjacent facets.

end

Compute $A_i = \frac{1}{3} \sum_j A_i^j$

Compute:

$$k_g(\mathbf{P}_i) = \frac{1}{A_i} (2\pi - \sum_j \theta_i^j)$$

$$k_m(\mathbf{P}_i) = \frac{1}{4A_i} \sum_j \gamma_i^j \|\mathbf{e}_i^j\|$$

Compute $k_1(\mathbf{P}_i)$ and $k_2(\mathbf{P}_i)$ by solving the quadratic equation given by the relationships $k_1 k_2 = k_g$ and $(k_1 + k_2)/2 = k_m$.

Compute the associated principal directions $\mathbf{t}_1(\mathbf{P}_i)$ and $\mathbf{t}_2(\mathbf{P}_i)$ by the least-square fitting method presented in [MEY 03], section 5.3.

end

Algorithm 3: Computing differential parameters by using discrete differential geometry operators.

Input: Vertex \mathbf{P}_i , normal \mathbf{n}_i

Output: Principal curvatures $k_1(\mathbf{P}_i)$ and $k_2(\mathbf{P}_i)$ and the associated principal directions $\mathbf{t}_1(\mathbf{P}_i)$ and $\mathbf{t}_2(\mathbf{P}_i)$

begin

forall couples of neighbor vertices $(\mathbf{P}_j, \mathbf{P}_k) \in N(\mathbf{P}_i)$ **do**

Compute the “oppositeness” value $M = \mathbf{P}_j \mathbf{P}_i \cdot \mathbf{P}_i \mathbf{P}_k$. The greater is M , the more opposite are \mathbf{P}_j and \mathbf{P}_k with respect to \mathbf{P}_i .

Sort the couples in the decreasing order of oppositeness and keep the best couples in a list Λ .

foreach couple $(\mathbf{P}_j, \mathbf{P}_k)$ of Λ **do**

Compute the circle \mathcal{C}_{jk} passing through $(\mathbf{P}_j, \mathbf{P}_i, \mathbf{P}_k)$

Compute its center \mathbf{C}_{jk} .

$\mathbf{n}_{jk} = \mathbf{C}_{jk} \mathbf{P}_i / \|\mathbf{C}_{jk} \mathbf{P}_i\|$ is the normal vector to \mathcal{C}_{jk} at \mathbf{P}_i .

$k_{jk} = 1 / \|\mathbf{C}_{jk} \mathbf{P}_i\|$ is the curvature of \mathcal{C}_{jk} at \mathbf{P}_i

Compute $\mathbf{t}_{jk} = \mathbf{n}_i \times \mathbf{n}_{jk}$. By definition, \mathbf{t}_{jk} is a vector orthogonal to \mathbf{n}_i and is then a tangent vector.

Let us define Π_{jk} the normal plane at \mathbf{P}_i in the direction of the tangent vector \mathbf{t}_{jk} . The plane containing \mathcal{C}_{jk} can be considered as a rotation of Π_{jk} around the axis $(\mathbf{P}, \mathbf{t}_{jk})$ by an angle α_{jk} given by $\cos \alpha_{jk} = \mathbf{n}_i \cdot \mathbf{n}_{jk}$

We can then apply Meusnier’s theorem (see equation [1.26]) to get an approximation of the normal curvature in the direction \mathbf{t}_{jk} : $k_{\mathbf{t}}^{jk} = k_{jk} \cos \alpha_{jk}$

end

Now, we can use the Euler’s theorem to estimate the principal curvatures. We have (see equation [1.24]):

$$k_1(\mathbf{P}_i) \cos^2 \theta_{jk} + k_2(\mathbf{P}_i) \sin^2 \theta_{jk} = k_{\mathbf{t}}^{jk} \text{ where } \theta_{jk} = \angle(\mathbf{t}_1, \mathbf{t}_{jk})$$

In any other tangent frame shifted by an angle θ_0 with respect to \mathbf{t}_1 , we can write:

$$k_1(\mathbf{P}_i) \cos^2(\theta_{jk} - \theta_0) + k_2(\mathbf{P}_i) \sin^2(\theta_{jk} - \theta_0) = k_{\mathbf{t}}^{jk}$$

If we develop, we obtain:

$$a \cos^2 \theta_{jk} + b \cos \theta_{jk} \sin \theta_{jk} + c \sin^2 \theta_{jk} = k_{\mathbf{t}}^{jk}$$

If we have n couples $(\mathbf{P}_j, \mathbf{P}_k)$, we get n equations. By using a least square method, we can compute a , b and c and then θ_0 . It gives directly the frame of the principal directions $\mathbf{t}_1(\mathbf{P}_i)$ and $\mathbf{t}_2(\mathbf{P}_i)$ and the principal curvatures $k_1(\mathbf{P}_i)$ and $k_2(\mathbf{P}_i)$.

end

end

Algorithm 4: Computing differential parameters by integrating 2D curvatures.

1.3.6. Integrating 2D curvatures

One of the first algorithms to compute differential parameters on a 3D mesh was described in [CHE 92]. It is based on the computation of the radii of circles going through the considered vertex and two neighbor vertices. For each circle, the inverse of the radius gives an estimation of the curvature in a tangent direction thanks to Meusnier's theorem. Based on the estimation for several circles, it becomes possible to estimate principal curvatures thanks to Euler's theorem. A version of the algorithm is given below:

Note that, in the original method, the normal \mathbf{n}_i is computed based on the vectors \mathbf{n}_{jk} .

In [WAT 01] and [DON 05], we find other methods which are also based on Euler's theorem. In both cases, approximation of the normal curvatures is based on only one neighbor \mathbf{P}_j , but it involves the normal vector \mathbf{n}_j .

1.3.7. Tensor of curvature: Taubin's formula

In this method, the idea is to compute the tensor of curvature, which is the map that associates with each point \mathbf{P} of a surface Σ , a 3×3 matrix that measures the directional curvature $k_{\mathbf{t}}$ for any tangent vector \mathbf{t} . In [TAU 95], the author proposed to use the following formula:

$$M(\mathbf{P}) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} k_{\mathbf{t}}(\theta) \mathbf{t}(\theta) \cdot \mathbf{t}^t(\theta) d\theta \quad [1.27]$$

with the notations $k_{\mathbf{t}}(\theta)$ and $\mathbf{t}(\theta)$ introduced in section 1.2.6.

If we write the above equation in the frame $(\mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$ defined by the principal directions and the normal at \mathbf{P} and if we use Euler's theorem, we get:

$$M(\mathbf{P}) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} (k_1 \cos^2 \theta + k_2 \sin^2 \theta) \begin{pmatrix} \cos \theta \\ \sin \theta \\ 0 \end{pmatrix} (\cos \theta \quad \sin \theta \quad 0) d\theta$$

which results in:

$$M(\mathbf{P}) = \begin{pmatrix} \frac{3}{8}k_1 + \frac{1}{8}k_2 & 0 & 0 \\ 0 & \frac{1}{8}k_1 + \frac{3}{8}k_2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Input: Vertex \mathbf{P}_i , normal vector \mathbf{n}_i

Output: Principal curvatures $k_1(\mathbf{P}_i)$ and $k_2(\mathbf{P}_i)$ with the associated principal directions $\mathbf{t}_1(\mathbf{P}_i)$ and $\mathbf{t}_2(\mathbf{P}_i)$.

begin

forall *neighbor vertex* $\mathbf{P}_j \in N(\mathbf{P}_i)$ **do**

 Compute the unit tangent vector \mathbf{t}_i^j at \mathbf{P}_i in the direction of \mathbf{P}_j .

 For this, project and normalize the vector $\mathbf{P}_i\mathbf{P}_j$ on the tangent plane at \mathbf{P}_i which is defined by \mathbf{n}_i .

 If we work in the normal plane at \mathbf{P}_i containing \mathbf{P}_j , we can write the coordinates of \mathbf{P}_j as $y = \mathbf{n}_i(\mathbf{P}_j - \mathbf{P}_i)$ and $x = \mathbf{t}_i^j(\mathbf{P}_j - \mathbf{P}_i)$.

 By applying an approximation of equation [1.25], we can write:

$$k(\mathbf{P}_j) = \frac{2 \mathbf{n}_i(\mathbf{P}_j - \mathbf{P}_i)}{\|\mathbf{t}_i^j(\mathbf{P}_j - \mathbf{P}_i)\|^2}$$

 Note that in Taubin's paper, the approximation is slightly different as it is based on a circle interpolation.

 Compute the area a_i^j of the two adjacent facets.

end

The discrete formulation of the tensor of curvature at \mathbf{P}_i is:

$$M(\mathbf{P}_i) = \frac{1}{\sum_j a_i^j} \sum_{\mathbf{P}_j} a_i^j k(\mathbf{P}_j) \mathbf{t}_i^j \cdot \mathbf{t}_i^{j^t}$$

Extract the non-null eigenvectors with the corresponding eigenvalues λ_1, λ_2 of the 3×3 tensor $M(\mathbf{P}_i)$ (see [TAU 95] for details on the numerical algorithm).

The principal directions $\mathbf{t}_1(\mathbf{P}_i)$ and $\mathbf{t}_2(\mathbf{P}_i)$ are inferred from the eigenvectors and the principal curvatures are given by:

$$k_1(\mathbf{P}_i) = 3\lambda_1 - \lambda_2$$

$$k_2(\mathbf{P}_i) = 3\lambda_2 - \lambda_1$$

end

Algorithm 5: Computing differential parameters by using Taubin's tensor of curvature.

In this diagonal matrix, we have two non-zero eigenvalues $\lambda_1 = \frac{3}{8}k_1 + \frac{1}{8}k_2$ and $\lambda_2 = \frac{1}{8}k_1 + \frac{3}{8}k_2$, which are associated with the eigenvectors \mathbf{t}_1 and \mathbf{t}_2 . We then get:

$$k_1 = 3\lambda_1 - \lambda_2 \quad k_2 = 3\lambda_2 - \lambda_1$$

with the normal vector \mathbf{n} being the third eigenvector associated with the eigenvalue 0.

Taubin then proposed to approximate, for any vertex \mathbf{P}_i of a 3D mesh, the matrix $M(\mathbf{P}_i)$. For this, he discretized the circular area around \mathbf{P}_i by a polygon linking the neighbor vertices \mathbf{P}_j of \mathbf{P}_i . This leads to a discrete formulation of equation [1.27], which allows us to compute the differential parameters by the following algorithm:

1.3.8. Tensor of curvature based on the normal cycle theory

In [COH 03], the authors proposed another definition of the curvature tensor, which has theoretical foundations as well as some convergence properties.

For a vertex \mathbf{P}_i of the 3D mesh that we assume triangular, we can take all the vertices \mathbf{P}_j of the neighborhood $N(\mathbf{P}_i)$. This defines the set of connecting edges \mathbf{e}_i^j , which are defined by their unit vectors \mathbf{t}_i^j (so with the same direction as \mathbf{e}_i^j). For an edge \mathbf{e}_i^j , we can define the following 3×3 matrix $M(\mathbf{e}_i^j)$ as:

$$M(\mathbf{e}_i^j) = \frac{1}{B_i^j} \|\mathbf{e}_i^j\| \gamma_i^j \mathbf{t}_i^j \cdot \mathbf{t}_i^{j^t}$$

where B_i^j is the half area of the two adjacent triangles incident to the edge \mathbf{e}_i^j and γ_i^j is the angle between the normal vectors to these triangles. The sign of γ_i^j is chosen to be positive if the two triangles form a convex dihedron and negative if it is concave.

By taking the approximation of the curvature of the edge \mathbf{e}_i^j by a cylinder portion (see section 1.3.5 and Figure 1.5), we have $k_1 = \gamma_i^j \|\mathbf{e}_i^j\| / A_i^j$ and $k_2 = 0$ where A_i^j is the area of the cylinder portion. The principal directions

corresponding to k_1 and k_2 are respectively given by a vector \mathbf{v}_i^j orthogonal to \mathbf{t}_i^j and tangent to the cylinder and by \mathbf{t}_i^j .

Now, if we write $M(\mathbf{e}_i^j)$ in the orthonormal frame $(\mathbf{t}_i^j, \mathbf{v}_i^j, \mathbf{n}_i^j)$ where $\mathbf{n}_i^j = \mathbf{t}_i^j \times \mathbf{v}_i^j$, we get:

$$M(\mathbf{e}_i^j) = \frac{1}{B_i^j} \|\mathbf{e}_i^j\| \gamma_i^j \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$

If we assimilate B_i^k and A_i^k , it results in:

$$M(\mathbf{e}_i^j) = \begin{pmatrix} \frac{A_i^j}{B_i^j} k_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \approx \begin{pmatrix} k_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

As the matrix is written under a diagonal form, this means that we have an eigenvalue equal to k_1 , associated with the eigenvector \mathbf{t}_i^j (which is the principal direction corresponding to k_2), a null eigenvalue which can be related to k_2 , associated with \mathbf{v}_i^j (which is the principal direction corresponding to k_1) and another null eigenvalue associated with \mathbf{n}_i^j .

This means that the principal curvatures (k_1, k_2) and their associated principal directions can be defined as eigenvalues and eigenvectors of $M(\mathbf{e}_i^j)$. Note that the principal directions are switched with respect to the principal curvatures.

Now, the idea is to integrate all tensors $M(\mathbf{e}_i^j)$ in a unique tensor centered in \mathbf{P}_i :

$$M(\mathbf{P}_i) = \frac{1}{B_i} \sum_{\mathbf{P}_j \in N(\mathbf{P}_i)} \|\mathbf{e}_i^j\| \gamma_i^j \mathbf{t}_i^j \cdot \mathbf{t}_i^{j^t}$$

where $B_i = \sum_j B_i^j$ if the area of the sub-mesh formed by all the vertices belonging to $N(\mathbf{P}_i)$.

If we generalize the formulas obtained with $M(\mathbf{e}_i^j)$, we may assume that we can compute the differential parameters at vertex \mathbf{P}_i based on the

eigenvalues and eigenvectors of $M(\mathbf{P}_i)$. More precisely, there will be always a null eigenvalue corresponding to the normal vector \mathbf{n}_i , the non-null eigenvalues will correspond to the principal curvatures and the associated eigenvectors will be the principal directions. In fact, all these results can be formally demonstrated by using a mathematical theory called *normal cycle* [COH 03], which provides a unified way to define curvature in both smooth and polyhedral surfaces.

A simplified version of the method can be implemented by the following algorithm:

Input: Vertex \mathbf{P}_i , normal vector \mathbf{n}_i
Output: Principal curvatures $k_1(\mathbf{P}_i)$ and $k_2(\mathbf{P}_i)$ with the associated principal directions $\mathbf{t}_1(\mathbf{P}_i)$ and $\mathbf{t}_2(\mathbf{P}_i)$.

begin

forall neighbor vertex $\mathbf{P}_i^j \in N(P_i)$ **do**

 Compute the unit tangent vector \mathbf{t}_i^j at \mathbf{P}_i in the direction of \mathbf{P}_j .
 For this, project and normalize the vector $\mathbf{P}_i\mathbf{P}_j$ on the tangent plane at \mathbf{P}_i defined by \mathbf{n}_i .
 Compute the half-area a_i^j of the two adjacent facets.

end

The discrete formulation of the tensor of curvature at \mathbf{P}_i is:

$$M(\mathbf{P}_i) = \frac{1}{\sum_j a_i^j} \sum_{\mathbf{P}_j \in N(P_i)} \|\mathbf{e}_i^j\| \gamma_i^j \mathbf{t}_i^j \cdot \mathbf{t}_i^{j^t}$$

Extract the two eigenvectors \mathbf{v}_1 and \mathbf{v}_2 corresponding to the two maximum eigenvalues λ_1 and λ_2 .
The couples $(\lambda_2, \mathbf{v}_1)$ and $(\lambda_1, \mathbf{v}_2)$ correspond to the principal directions and curvatures $(k_1(\mathbf{P}_i), t_1(\mathbf{P}_i))$ and $(k_2(\mathbf{P}_i), t_2(\mathbf{P}_i))$ (beware that the eigenvectors are switched with respect to the eigenvalues).

end

Algorithm 6: Computing differential parameters by using the normal cycle based tensor of curvature.

A demonstration application of this method, with the C++ source, is available on the Web³.

³ <http://www-sop.inria.fr/members/Pierre.Alliez/demos/curvature/>.

1.3.9. Integral estimators

At a point \mathbf{P} of a closed surface Σ , let us define the ball $B(\mathbf{P}, r)$ of radius r . If we assume that, at any point, the normal \mathbf{n} will be outward-pointing (i.e. pointing towards the exterior of the volume enclosed by the surface), we can define $B_{\Sigma}^{+}(\mathbf{P}, r)$ as the sub-volume of $B(\mathbf{P}, r)$, which is above the surface Σ (i.e. on the side containing the normal vector \mathbf{n}) (see Figure 1.6).

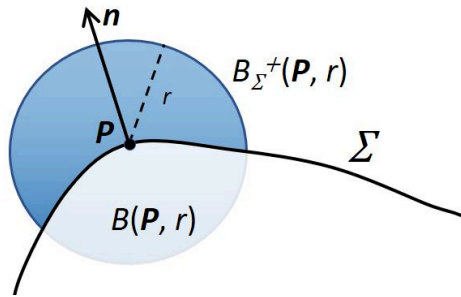


Figure 1.6. $B(\mathbf{P}, r)$ is the ball centered at \mathbf{P} of radius r . $B_{\Sigma}^{+}(\mathbf{P}, r)$ (in dark blue) is the sub-volume of $B(\mathbf{P}, r)$ which is above the surface Σ , i.e. on the side containing the normal vector \mathbf{n} which points outside the surface

In [POT 07], the authors propose to define the differential parameters with respect to the volume integral of some functions. For this purpose, the authors work in the frame $(\mathbf{P}, \mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$, from which we can write the quadratic approximation of the surface Σ around \mathbf{P} (see section 1.2.8):

$$z(x, y) = \frac{1}{2}(k_1 x^2 + k_2 y^2)$$

For any function f , its integral value $I(f, r)$ over the sub-volume $B_{\Sigma}^{+}(\mathbf{P}, r)$ can be computed as (see Figure 1.7):

$$I(f, r) = \int_{B(\mathbf{P}, r)^+} f(x, y, z) \, dx \, dy \, dz = I_1(f, r) + I_2(f, r)$$

where:

– $B(\mathbf{P}, r)^+$ is the half-ball above the tangent plane;

– I_1 is the integral of f in the volume between the tangent plane and the surface Σ . Note that the sign minus before I_1 is due to the sign of z , which is directly related to the orientation of \mathbf{n} . For example, in Figure 1.7, z and then I_1 will be negative;

– I_2 is the small volume between the tangent plane and the surface Σ , which does not belong to $B(\mathbf{P}, r)^+$, so that it is outside the sphere $x^2 + y^2 + z^2 = r^2$ but inside the cylinder $x^2 + y^2 = r^2$. The plus sign before I_2 is also related to the predefined orientation of \mathbf{n} .

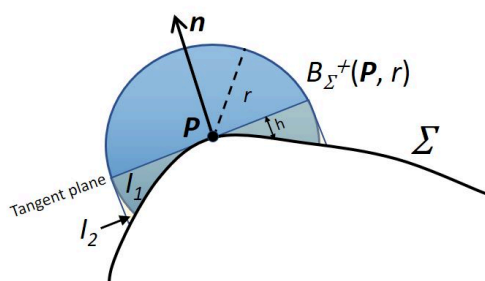


Figure 1.7. The volume $B_{\Sigma}^{+}(\mathbf{P}, r)$ is the sum of the half ball which is above the tangent plane (in dark blue), of the volume I_1 between the tangent plane and the surface Σ minus the small volume I_2 between the tangent plane and the surface Σ which does not belong to $B(\mathbf{P}, r)^+$ that is outside the sphere $x^2 + y^2 + z^2 = r^2$ but inside the cylinder $x^2 + y^2 = r^2$

This formula is valid whatever the point \mathbf{P} is, elliptical, hyperbolic or parabolic. We can show that I_2 is negligible in the computation of $I(f, r)$ and we can approximate I_1 as:

$$I_1(f, r) \approx \int_{x^2+y^2 \leq r^2} \left(\int_{z=0}^{\frac{1}{2}(k_1x^2+k_2y^2)} f(x, y, z) dz \right) dx dy$$

This leads to the approximation of the integral estimator $I(f, r)$:

$$I(f, r) \approx \int_{B(\mathbf{P}, r)^+} f(x, y, z) dx dy dz - \int_{x^2+y^2 \leq r^2} \left(\int_{z=0}^{\frac{1}{2}(k_1x^2+k_2y^2)} f(x, y, z) dz \right) dx dy \quad [1.28]$$

We can use this formula to measure approximately the volume $V(\mathbf{P}, r)$ of $B_{\Sigma}^+(\mathbf{P}, r)$ by using the function $f(x, y, z) = 1$. A conversion to polar coordinates leads to (with $k_m = 1/2(k_1 + k_2)$):

$$V(\mathbf{P}, r) = \frac{2\pi}{3}r^3 - \frac{\pi}{4}k_m r^4$$

We can also get an approximation of the coordinates of the barycenter \mathbf{G} of the sub-volume $B_{\Sigma}^+(\mathbf{P})$ by using the functions $f(x, y, z) = x, y$ or z and we get:

$$\mathbf{G} \left(0, 0, \frac{3}{8}r + \frac{9}{64}k_m r^2 \right)$$

We can now define the covariance matrix:

$$J(\mathbf{P}, r) = \int_{B_{\Sigma}^+(\mathbf{P}, r)} (\mathbf{X} - \mathbf{G})(\mathbf{X} - \mathbf{G})^t d\mathbf{X}$$

As there is symmetry with respect to the planes (\mathbf{G}, x, z) and (\mathbf{G}, y, z) , the non-diagonal terms are null and we can simplify the expression in:

$$J(\mathbf{P}, r) = \int_{B_{\Sigma}^+(\mathbf{P}, r)} \mathbf{X}\mathbf{X}^t d\mathbf{X} - V(\mathbf{P}, r)\mathbf{G}\mathbf{G}^t$$

that we can approximate by:

$$\begin{aligned} J(\mathbf{P}, r) &\approx \text{diag} (I(x^2, r), I(y^2, r), I(z^2, r)) \\ &\quad - \text{diag} \left(0, 0, \frac{\pi}{4}r^4 \left(\frac{3}{8}r + \frac{9}{64}k_m r^2 \right) \right) \end{aligned}$$

By converting in polar coordinates, we can easily compute $I(x^2, r)$, $I(y^2, r)$ and $I(z^2, r)$. Then by developing the expression of $J(\mathbf{P}, r)$, we find the three diagonal terms. The first two correspond to the two eigenvalues associated with the eigenvectors which are in the plane (\mathbf{B}, x, y) . They are given by:

$$\begin{aligned} m_1 &= \frac{2\pi}{15}r^5 - \frac{\pi}{48}(3k_1 + k_2)r^6 \\ m_2 &= \frac{2\pi}{15}r^5 - \frac{\pi}{48}(k_1 + 3k_2)r^6 \end{aligned}$$

These formulas then allow us to compute the principal curvatures k_1 and k_2 by using the following algorithm:

Input: Vertex \mathbf{P}_i , radius r

Output: Principal curvatures $k_1(\mathbf{P}_i)$ and $k_2(\mathbf{P}_i)$

begin

Transform the 3D mesh into a binary 3D image by computing an occupancy voxel grid via a scan conversion algorithm (for fundamentals of such algorithm, see the seminal paper of [KAU 87]). Voxel value is 1 if it is inside the surface and 0 if outside.

The voxels belonging to $B_{\Sigma}^+(\mathbf{P}_i, r)$ are those which are labeled 1 and whose distance of their center to the vertex \mathbf{P}_i is inferior to r .

Compute $J(\mathbf{P}_i, r)$ based on the voxels of $B_{\Sigma}^+(\mathbf{P}_i, r)$. This is equivalent to perform a Principal Component Analysis of the centers of these voxels.

This step can be transformed into a convolution expression, which means that Fast Fourier Transform can be employed for efficiency [POT 07].

Note that in [POT 09], it is proposed to use an octree structure instead of a binary image which allows us to perform computations based on the exact coordinates of the mesh vertices and not on the centers of discrete voxels.

Extract the two eigenvalues m_1 and m_2 of $J(\mathbf{P}_i, r)$ which correspond to the two eigenvectors close to the tangent surface (i.e. orthogonal to \mathbf{n}).

$$k_1(\mathbf{P}_i) = \frac{6}{\pi r^6}(m_2 - 3m_1) + \frac{8}{5r} \quad k_2(\mathbf{P}_i) = \frac{6}{\pi r^6}(m_1 - 3m_2) + \frac{8}{5r}$$

The principal directions can be estimated by projecting the two corresponding eigenvectors on the tangent plane. Notice that they are not necessarily orthogonal.

end

Algorithm 7: Computing differential parameters by local integral estimators.

An implementation of another algorithm [COE 14], which is also based on integral invariants can be found in the collaborative project DGtal⁴ developed as an open-source C++ library.

⁴ <http://dgtal.org/doc/0.9.2/moduleIntegralInvariant.html>.

1.3.10. Processing unstructured 3D point clouds

If we now have only an unstructured 3D cloud of points \mathbf{P}_i without any edge or facet as in a mesh, how can we compute differential parameters?

Input: Vertex \mathbf{P}_i

Output: Principal curvatures $k_1(\mathbf{P}_i)$ and $k_2(\mathbf{P}_i)$ with the associated principal directions $\mathbf{t}_1(\mathbf{P}_i)$ and $\mathbf{t}_2(\mathbf{P}_i)$.

begin

Estimate the normal vector \mathbf{n}_i at point \mathbf{P}_i (we can find a reference to a method at the end of section 1.3.3). The plane passing by \mathbf{P}_i and orthogonal to \mathbf{n}_i can be considered as the tangent plane to \mathbf{P}_i . We can then define an orthogonal frame (\mathbf{u}, \mathbf{v}) in this tangent plane.

Decompose this tangent plane into six 60° slices around \mathbf{P}_i .

For each slice j , find the point of the cloud \mathbf{P}_i^j whose projection on the slice is the closest to \mathbf{P}_i . If this projected point is not within a predefined distance, \mathbf{P}_i^j will not be taken into account in the following steps.

Estimate the normal vectors \mathbf{n}_i^j at points \mathbf{P}_i^j

Compute for all pairs of points the positional variation

$$\Delta \mathbf{P}_i^{jj'} = \mathbf{P}_i^{j'} - \mathbf{P}_i^j \text{ and the normal variation } \Delta \mathbf{n}_i^{jj'} = \mathbf{n}_i^{j'} - \mathbf{n}_i^j.$$

After projection in the tangent plane frame (\mathbf{u}, \mathbf{v}) , we can estimate the shape operator by (see equations [1.18]):

$$\begin{pmatrix} \Delta \mathbf{n}_i^{jj'} \cdot \mathbf{u} \\ \Delta \mathbf{n}_i^{jj'} \cdot \mathbf{v} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \Delta \mathbf{P}_i^{jj'} \cdot \mathbf{u} \\ \Delta \mathbf{P}_i^{jj'} \cdot \mathbf{v} \end{pmatrix} \quad [1.29]$$

Given more than three pairs of points, we obtain an over-constrained system which can be solved by a least-squares method.

Once we get the coefficients (a, b, c, d) of the shape operator matrix, we can compute its eigenvalues and eigenvectors to obtain the principal curvatures and the associated principal directions (see section 1.2.4).

An iteration process coupled with robust estimators can refine the estimation of normal variation and then the principal curvature values.

end

Algorithm 8: Computing differential parameters in the case of an unstructured 3D point cloud by the method [KAL 09].

As we have no more connection information between the points \mathbf{P}_i , the definition of the normal vector \mathbf{n}_i or of the point neighborhood $N(\mathbf{P}_i)$ is no more valid, and the methods based on local fitting, integration of 2D curvatures or tensor of curvatures (see respectively sections 1.3.4, 1.3.6, 1.3.7 and 1.3.8) cannot be used. Moreover, we have no facet so the method based on discrete differential geometry operators (see section 1.3.5) becomes inapplicable. At last, we have no mesh surface to define outside and inside which prevents using a method based on integral estimators (see section 1.3.9).

Of course, we could reconstruct a 3D mesh based on the points \mathbf{P}_i (for a review of reconstruction methods, see [BER 14]) and then apply one of the algorithms seen in the previous sections. But some specific methods have been developed to compute differential parameters in the case of an unstructured 3D point cloud. In the following, we describe the algorithm proposed in [KAL 09] (see also some details in [KAL 07b]).

Note that this method can also be used for 3D meshes [KAL 07b]. An implementation is publicly available as a Windows executable⁵.

We can also find in the Point Cloud Library⁶ an implementation of another method to estimate principal directions and curvatures of a 3D point cloud based on the principal component analysis of the normal vectors.

1.3.11. Discussion of the methods

Convergence towards theoretical values

The 3D mesh \mathcal{M} is a discrete representation of an unknown continuous surface \mathcal{S} . Thus, a natural question is to assess if the differential parameters computed by the methods described in the previous sections converge towards the theoretical values obtained by explicit formulas (see section 1.2.5) when the discretization becomes “finer”.

We could think that the convergence is only related to the local vertex density. But, in [BOR 03], it is shown that the angle defect, used in the DDG operator to compute the Gaussian curvature, depends also on the vertex

⁵ <http://people.cs.umass.edu/kalo/papers/curvature/index.html>.

⁶ http://docs.pointclouds.org/trunk/classpcl_1_1_principal_curvatures_estimation.html.

valence. Moreover, in [RUS 04], the author describes some vertex arrangements that produce large errors in the normal cycle-based tensor method, whatever the density vertex is (note that the author also proposed a method to solve this problem with an implementation in C++ which is publicly available in the trimesh2 library⁷).

More generally, in [XU 05] the authors describe a counterexample mesh that prevents any Gaussian curvature and mean curvature method to converge. Based on the vertices of the mesh, they define different continuous surfaces by a quadratic approximation. This leads to different potential continuous curvature values which never correspond to the unique result given by a discrete method.

Nevertheless, it is proved in [LIU 07] that when \mathcal{M} is a quadrilateral mesh, the Gaussian curvature computed with the DDG method has a quadratic convergence rate under a "parallelogram criterion". Moreover, when the quadrilateral mesh is based on a discrete net of curvature lines [BAU 10], the discrete estimation (by a specific described method) of the principal curvatures converges to the exact values when the distances between vertices become smaller. However, in [XU 09], it is emphasized that, in the general case, there is no convergence for a regular vertex with valence 4.

When \mathcal{M} is a Delaunay triangulation restricted to the surface and under a local uniformity condition on the sampling, the authors of [COH 03] proved that the normal cycle-based curvature estimator converges linearly with respect to the sampling density. This case is important in practice as several surface reconstruction algorithms are based on the Delaunay triangulation of the input points.

A general convergence property can be found in [CAZ 05a], where it is proved that a polynomial fitting of degree d estimates the principal curvatures and directions (away from umbilics) to accuracy $O(h^{d-1})$, where h is the sampling density. However, in fact, this property depends on some conditions on the positions of points. They are not explicitly given in the paper but as in the counterexample described before, they are linked to the unicity of the approximation of the vertices by the polynomial function. In [XU 13], we can find a well-posedness condition (which is considered by the authors as a very

⁷ <http://gfx.cs.princeton.edu/proj/trimesh2/>.

mild one) which ensures that the Gaussian and mean curvatures computed by a quadratic local fitting have a linear convergence rate towards the continuous values.

Robustness to noise

In many cases, the 3D mesh \mathcal{M} is obtained by scanning a physical object. This requires at least three steps: acquisition where the sensor takes measurements of points with a limited resolution (see, for example, [SAN 09] for a review of the available techniques), surface reconstruction which builds a 3D mesh by connecting or interpolating points based on some regularity constraints [BER 14] and post-processing in order to smooth or “repair” the 3D mesh [WEY 04]. This processing pipeline introduces some errors or inaccuracies in the coordinates of vertices or in the mesh connectivity, which both form the so-called “noise”. In methods to compute differential parameters, the problem is then to cope with these distortions, i.e. to be robust to noise.

The first idea is to optimize the algorithm parameters as the neighborhood size for the local fitting and the tensor of curvature methods or the radius of the ball for the integral estimators. In general, by increasing these values, we average the measures and then, we diminish the influence of the noise.

The second idea consists of not taking into account all the measurements computed around \mathbf{P}_i with the same reliability. For example, if we consider that the further a vertex \mathbf{P}_j is from \mathbf{P}_i , the more inconsistent are its associated parameters (as the normal vector), we can introduce weights based on the distance $\|\mathbf{P}_i\mathbf{P}_j\|$. This is particularly useful when we sum some values over the neighbor vertices \mathbf{P}_j as in the local fitting and tensor of curvature methods. Another way is to use some robust estimators [ROU 87] as the least trimmed squares in the minimization of E in the local fitting method.

In fact, both ideas are often combined. For example, in [KAL 09] (see also the Windows executable⁸), the authors propose to use a neighborhood radius of three times the average distance of \mathbf{P}_i to its direct neighbors, and at the same time, they implement an Iteratively Reweighted Least Squares approach to weight the contribution of each neighbor.

⁸ <http://people.cs.umass.edu/kalo/papers/curvature/index.html>.

A last idea is to apply some smoothing process on the 3D mesh before computing the differential parameters. But the risk is then to make some geometric features to disappear as, for example, the ridges. Indeed, comparing the values of the curvatures before and after smoothing appears to be a good method to measure locally the roughness of the 3D mesh (see [LAV 09b] for more details).

Multiscale approach

Several methods propose a multiscale approach. It consists of defining a series of n values between a low and a high bound. Each value corresponds to a *scale* and is used to compute differential parameters. We then get a series of results for principal curvatures and directions, which are fused in definitive results by taking into account their scales.

These n values can be used directly in a method. For example, they can represent neighborhood sizes for local fitting methods (see, for example, [PAN 10b]) or radii of the ball for integral estimators (as in [YAN 06]). Another way is to use these n values to smooth the mesh \mathcal{M} in order to build different versions, which will present different levels of detail according to the scale. We can then apply a method with fixed parameters to all these smoothed versions [MOK 01] and select the result given by the most adapted scale or combine the results obtained at different scales.

The main problem is to define the bound values. For example, if the value is too low, the neighborhood or the ball may be too small, too few vertices will be taken into account and the computed curvatures will be inaccurate. If the value is too high, the ball may be too large and may incorporate some parts of the mesh which are no more connected to the considered vertex (see [YAN 06], Figure 4) or the smoothing process will be so important that important geometric details of the mesh will be definitively lost. Very often, the bound values are defined proportionally to the average edge length in the mesh. But this solution is not adequate when the triangle size varies a lot and this is frequent when the mesh was reconstructed from a scan of a real object.

In [SEE 16], the authors propose an algorithm to find an optimal scale (in this case, the scale corresponds to the radius of the ball for the integral estimator method) in an automatic way. The scales are defined independently for each vertex \mathbf{P}_i in order to cope with local variation of the edge length. A first scale is defined by averaging the lengths of all the edges connected to \mathbf{P}_i .

This value is then iteratively modified by taking into account the values computed for the neighboring vertices and this gives the lower scale $s_0(\mathbf{P}_i)$. The upper scale is computed by $s_n(\mathbf{P}_i) = s_0(\mathbf{P}_i) \cdot f^i$, where f is a float value greater than 1.0 and i an integer. i is set in order that $s_n(\mathbf{P}_i)$ corresponds to the radius of a ball of the same magnitude that the details of the surface we want to distinguish. For each scale $s \in [s_0, s_n]$, given the formulas of section 1.3.9, we compute the mean curvature $k_m^s(\mathbf{P}_i)$ which is a function of s . If \mathbf{P}_i is in a planar region, we have $k_m^s(\mathbf{P}_i) \approx 0$ and we define the optimal scale as the largest scale, which is below a given threshold. In the other cases, we draw the curve $k_m^s(\mathbf{P}_i)$ w.r.t. s , which presents in general one or more extrema. The optimal scale is then computed based on this (or these) extrema. If there is no extremum (except planar regions), the middle scale is selected. The authors show some examples where the local selection of the optimal scale allows us to take into account finer details of the mesh than with a fixed scale.

Comparison of the different methods

We have seen that many methods exist to compute differential parameters on a 3D mesh. Some of them have more theoretical foundations which give them some interesting properties of convergence or robustness; others are particularly adapted for a multi-scale approach. But is it possible to assess their practical efficiency?

In [GAT 06], the authors used different parametric surfaces for which they know the exact differential parameters (a sphere, a cylinder, a cubic polynomial or a sum of cosine and sine functions). Then, they introduce some distortions by tuning seven parameters in order to simulate different mesh resolutions, regularities, noises or valences. Methods of three classes – locally fitting parametric surfaces, DDG operators and tensor of curvatures – give results which are compared to the theoretical values. Among the conclusions, the authors emphasize that:

- some fitting methods (in particular cubic fitting) have better overall performance but at a quite important computational cost. The results are better with a large neighborhood;
- DDG methods are fast but they are more sensitive to valence, noise and mesh regularity;
- tensor of curvature method is based only on the vertex neighborhood and suffers from a severe sensitivity to noise normal to the surface.

In a similar manner, in [ANG 11], four methods belonging to the same three classes are tested on some parts of geometric primitives (a plane, a cylinder, a cone, a sphere and a torus). The authors introduce some noise in point coordinates and use different tessellations in order to analyze the sensitivity to mesh regularity and resolution. Their main conclusions are:

- all the methods have high sensitivity to noise if the neighborhood is restricted to the *1-ring*, i.e. the direct neighbors of the vertex;
- the local fitting method shows a quite general good capability whatever the vertex is, elliptical, hyperbolic, parabolic or an umbilic.

In [MAG 07], methods are tested on parametric NURBS surfaces and on scanned objects corresponding to geometric primitives (a plane, a sphere, a cylinder and a cone), which parameters were accurately measured. For each case, several meshes were produced corresponding to different resolutions, from about one hundred to several thousand triangles. Four different approaches – locally fitting a parametric surface, DDG, integrating 2D curvatures, tensor of curvatures – were selected and for each of them, a representative algorithm was implemented. By comparing the computed and theoretical values of the Gaussian and mean curvatures, the authors draw the following conclusions:

- DDG and surface fitting methods give best results on parametric and scanned data;
- on parametric data, DDG and surface fitting methods converge as the fineness of the mesh is improved;
- when resolution is very high in scanned objects, the relative error of the digitization process perturbs the accuracy of the computed curvatures.

In a recent paper [VÁŠ 16] (see also the companion paper [VÁŠ 17]), nine different methods belonging to locally fitting, DDG, normal cycle tensor of curvature and integral estimators classes are compared. They have all been implemented in C# in a framework which is publicly available⁹. The framework is modular and data, distortion sources, curvature estimators or evaluation routines can be added easily. Parametric surfaces, implicit functions

⁹ <http://graphics.zcu.cz/curvature.html>.

and NURBS surfaces have been discretized with different sampling schemes (e.g. rectangular, equilateral or random triangles) and different densities, and some noise was added. The computed and the theoretical curvatures were compared in order to assess the accuracy of the methods and the influence of their parameters. Main conclusions for noiseless meshes are:

- DDG methods provide the best results but only if the mesh sampling is regular;
- local fitting methods provide comparably good results even when the mesh regularity is lower;
- the normal cycle tensor method and the integral estimator provide, in general, results of considerably lower accuracy as they tend to smooth the results.

Whereas for noised meshes, we get:

- all the methods which are based on a small neighborhood (e.g. only the vertex neighbors) fail to provide any reasonable estimation of the curvatures. Even when a larger neighborhood is used, methods as local fitting give quite bad results;
- best results are obtained by integral estimators but only when an appropriate and quite large radius is used.

In fact, all these results demonstrate that no single estimator is efficient for all meshes. Thus, in [VÁŠ 16], the authors propose to construct a meta-estimator based on statistics collected during the evaluation of the different methods. This allows the meta-estimator to select one or several of them according to some properties of the input mesh (in particular, the "smoothness" defined by the discrete Laplacian operator applied on vertex positions). In fact, even a very simple meta-estimator which chooses between only two or three estimators improves considerably the average accuracy.

Note also that the efficiency of all methods depends of course on mesh preprocessing (in particular of surface smoothing) as well as on implementation details. For example, we can see in [WAN 09] that the accuracy and stability of the local fitting method (by polynomials) may be significantly changed by the choice of the numerical solver.

1.4. Feature line extraction

1.4.1. Introduction

We can mathematically define some lines on the surface to characterize its shape. Detecting such feature lines provide robust shape descriptors, relevant from a geometric and a topological point of view. This is essential in many applications such as remeshing [ALL 03], shape indexing [MAE 96], shape interrogation [HAH 08] or non-realistic rendering [RÖS 00a].

Similarly to [HOS 92], we can separate feature lines into three classes: the lines which are defined on the surface independently of its shape, the lines which exist only around some specific shape configurations and the lines which depend on some external parameters such as the view point or the direction of the incident light.

The first class of feature lines is very useful to parameter the entire surface in a way which is intrinsically adapted to its shape. It then becomes possible to analyze locally the surface and then to detect some patterns that defines a particular shape feature. In section 1.4.2, we will focus on the concept of “lines of curvature”.

In the second class of feature lines, we find in particular the lines which emphasize the salient parts of the surface which are very strong visual features of the shape. In section 1.4.3, we will describe the concept of “crest/ridge lines”.

In [HOS 92], we can find several examples of feature lines belonging to the third class. In particular, we find the *contour curves* which represents the intersection of the surface with equidistance parallel planes or the *highlight curves* which are the locus of points where the brightness (which is computed according to the ray source direction and the surface normal) to the observer’s eye is constant. In both cases, it requires to define a specific direction to determine the plane orientation or the view point. But in most of 3D modeling applications, we are reluctant to introduce such external parameters. As a result, we choose not to study this class of feature lines in this book.

1.4.2. Lines of curvature

Application to shape description

We have seen in section 1.2.3 the definition of the lines of curvature and how they form a network of orthogonal curves all over the surface, except at umbilics. This network can be used to define an orthogonal mesh composed of quadrilateral *principal patches* [MAR 83, SIN 90], which parameterize a surface in an intuitive or a useful manner for several applications.

Moreover, equation [1.15] shows that the lines of curvature characterize the maximum variation of the normal vector. As most of the lighting effects (shading, specularity) that give the perception of a surface are based on the variation of the normal vector, it is then possible to visualize the global shape of an object just by displaying the principal directions (see, for example, [GIR 00]) or the lines of curvature.

Thus, when drawing a concept sketch of an object, a 3D designer uses intuitively lines of curvature to emphasize the surface bending. Sometimes, this is accurate enough to get a very realistic view of the object by automatic shading and texturing, as proposed in [IAR 15]. More precisely, a skilled 3D designer aligns the edges of the objects with lines of curvature, whereas in spherical areas which correspond to umbilics, points will be sampled isotropically. By using a remeshing or quadrangulation algorithm driven by lines of curvature as proposed in [ALL 03, KÄL 07a, LI 11], we can reconstruct a very "efficient" 3D mesh, in the sense that it keeps the main geometric features of the initial mesh while it minimizes the number of faces. Lines of curvature can also be used to align strokes in automatic hatching techniques in order to display complex surfaces in a perceptually convincing way (see, for example, [RÖS 00a]).

Lines of curvature can also be integrated in a computer-aided design process. For instance, in [JOO 14], the authors show how the lines of curvatures of a curved surface can be used to define plates which can be locally adjusted in order to reconstruct this surface for shipbuilding or architectural free-form building applications. In [TSU 17], the authors describe a method to reconstruct from scanned 3D data the surface of objects which have been created by sweeping a specified 3D section curve along a 3D spine curve (as, for example, the hood of a car). The idea is to estimate a first surface and then compute its lines of curvature. The lines of curvature with

less torsion locally estimate the 3D spine curve. This makes possible to then reconstruct a definitive parametric surface which looks visually acceptable for designers. The method described in [TAK 16] maps the four sides of each principal patch onto a plane. It then connects these patches one by one by aligning the equilength adjacent edges using translations and rotations. These strips can then be assembled to form a 3D surface from which it is possible to build 3D objects with sheets of light material as paper or plastic. It is even possible to build complex and large objects with sheets of metal or carbon fiber-reinforced plastics by adding stiffeners or frames along the lines of curvature. Note also an application to architecture in [MES 18] where the authors propose a methodology to generate surfaces with planar lines of curvature. These surfaces called super-canal can then be easily built by using flat panels.

Computation of lines of curvature on parametric surfaces

Lines of curvature are not directly defined by an equation. They are in fact *integral curves* of the principal direction field which means that, at each point except umbilics, they are tangent to a principal direction.

Let us define the point $\mathbf{P}_0(u_0, v_0)$ on the surface Σ . We assume that it is not an umbilic and let $\mathcal{C}_i(u, v)$ ($i \in 1, 2$) be one of the two lines of curvature going through \mathbf{P}_0 and tangent to the principal direction corresponding to the principal curvature k_i . We now have to find all the points $\mathbf{P}(u, v)$ that define \mathcal{C}_i .

For this, we can follow the method proposed in [MAE 96] (see also [FAR 98] or section 9.4 of [PAT 10]). We will express the line of curvature as $u = u(s)$ and $v = v(s)$ where s is the arclength of \mathcal{C}_i from \mathbf{P}_0 . At each point \mathbf{P} , we can compute the coefficient $\gamma = dv/du$ by one of the two equations [1.9] or [1.10]:

$$\begin{cases} (M - k_i F) + \gamma(N - k_i G) = 0 \\ (L - k_i E) + \gamma(M - k_i F) = 0 \end{cases} \quad [1.30]$$

The first equation can be written as $\gamma = (k_i F - M)/(N - k_i G)$. Then, at a point $\mathbf{P}(u, v)$, we can introduce the scalar function $\alpha(\mathbf{P})$ which is non-null and write:

$$\frac{du}{ds} = \alpha(N - k_i G) \quad [1.31]$$

$$\frac{dv}{ds} = \gamma \frac{du}{ds} = \alpha(k_i F - M) \quad [1.32]$$

As we work with arclength parameterization, we have:

$$\left\| \frac{d\mathbf{P}}{ds} \right\|^2 = 1$$

which can be rewritten by using the first fundamental form (see equation [1.1]):

$$E \left(\frac{du}{ds} \right)^2 + 2F \left(\frac{du}{ds} \frac{dv}{ds} \right) + G \left(\frac{dv}{ds} \right)^2 = 1$$

By using equations [1.31] and [1.32], we find:

$$\alpha = \pm \frac{1}{\sqrt{E(N - k_i G)^2 + 2F(N - k_i G)(k_i F - M) + G(k_i F - M)^2}} \quad [1.33]$$

But if we use the second equation of 1.30, we have $\gamma = (k_i E - L)/(M - k_i F)$ and we get another formulation of α :

$$\alpha = \pm \frac{1}{\sqrt{E(M - k_i F)^2 + 2F(M - k_i F)(k_i E - L) + G(k_i E - L)^2}} \quad [1.34]$$

In conclusion, if we compute the coefficients E , F , G , L , M , N and the principal curvatures k_i at each point \mathbf{P} of the surface, it is possible to compute $\alpha(\mathbf{P})$ by one of the two above equations. We can then integrate equation [1.31] or [1.32] to define the two lines of curvature $\mathcal{C}_i(u, v)$.

In practice, the integration can be performed by standard numerical techniques such as Runge–Kutta (fixed step solver) or Adams (variable step solver). This allows us to compute quite precisely the lines of curvature of parametric surfaces such as an ellipsoid $P(\theta, \phi) = (a \cos \theta \cos \phi, b \cos \theta \sin \phi, c \sin \theta)$ [FAR 98], an elliptic paraboloid $P(u, v) = (u, v, au^2 + v^2)$ [JOO 14] or more generally a Bézier surface [JOO 14].

Accuracy of the lines of curvature depends on the number of integrated steps, but several issues arise:

– What formulation of α to use, either equation [1.33] or [1.34]? The idea proposed by [MAE 96] is to select the largest coefficients in equations [1.30] in order to avoid numerical inaccuracies. Since $(M - k_i F)$ is a common coefficient, equation [1.33] is used if $|N - kG| \leq |L + k_i E|$ and equation [1.34] otherwise. This prevents also using an equation with null coefficients.

– How to choose the sign of α ? This sign determines the direction along which the line of curvature is built. As emphasized in [MAE 96], choosing a fixed sign does not guarantee that the tangent vector:

$$\frac{d\mathbf{P}}{ds} = \frac{d\mathbf{P}}{du} \frac{du}{ds} + \frac{d\mathbf{P}}{dv} \frac{dv}{ds}$$

does not change direction along the computation of the line of curvature. An idea is then to integrate in the direction which is the closest to the direction of the tangent vector computed at the previous integration step. This leads to the following condition:

$$\left\| \frac{d\mathbf{P}}{ds} - \frac{d\mathbf{P}^p}{ds} \right\| < \left\| -\frac{d\mathbf{P}}{ds} - \frac{d\mathbf{P}^p}{ds} \right\|$$

where $\frac{d\mathbf{P}^p}{ds}$ is the tangent vector at the previous integration step. If this condition is not true, the sign of α has to be inverted for the integration.

– In some points which are not umbilics, the two coefficients of one of the equations [1.30] could be null, preventing using this equation to compute γ . But [FAR 98] emphasizes that this occurs if and only if the principal directions are tangent to the surface parameter lines. For general free-form surfaces, this occurs only at very isolated points and then has a little influence on the integration process.

– We have seen in section 1.2.3 that the lines of curvature are not defined at umbilic points. A problem then arises when a line of curvature passes through an umbilic. This corresponds to the fact that the point \mathbf{P} obtained after the integration step is such that $k_1(\mathbf{P}) = k_2(\mathbf{P})$. In this case, [MAE 96] proposes to slightly shift the position of \mathbf{P} . As for general free-form surfaces, umbilics are isolated points, this is a convenient way to avoid umbilic singularity and to draw long lines of curvature.

Computation of lines of curvature on a discrete 3D mesh

To compute the lines of curvature on a discrete 3D mesh, the authors of [RÖS 00a] propose to use a simplified version of the method described in the previous section. First, they compute the principal direction vectors \mathbf{t}_1 and \mathbf{t}_2 for all the vertices of the 3D mesh. This can be done by any of the methods described in section 1.3. They then propose to define the principal direction vectors at any point on a facet (assumed triangular) using a barycentric interpolation of the principal direction vectors computed at the vertices of the facet. It is then possible to integrate, facet after facet, the line of curvature from a vertex \mathbf{P}_0 along one of the two principal direction \mathbf{t} (\mathbf{t} corresponds either to \mathbf{t}_1 or \mathbf{t}_2) (see algorithm 9):

Nevertheless, this algorithm gives only a part of the line of curvature, L^+ . We then have to run the algorithm again from \mathbf{P}_0 but along the opposite vector $-\mathbf{t}$ to get the second part L^- . The line of curvature L going through \mathbf{P}_0 is then obtained by merging L^+ and L^- .

We also face some difficulties which are similar to the continuous case. First, we must be sure that all the principal direction vectors of the facet are oriented consistently in order that the barycentric interpolation gives a significant result. For this, it is proposed in [RÖS 00a] that when a segment $\mathbf{P}_i\mathbf{P}_{i+1}$ enters a new facet F_{i+1} , the principal direction vectors of the three vertices of the facet are oriented in order that their dot product with $\mathbf{P}_i\mathbf{P}_{i+1}$ is maximum.

The step value h has a large influence on the integration of the line of curvature. In particular, if it is too large, we can easily deviate from the actual line. The idea is then to adjust it adaptively at each step, in particular with respect to the shape of the current facet F_i . We can find in [RÖS 00a], the following formula which assumes that the facets are triangles. It is heuristic but works well according to many experiments and is very fast to compute:

$$h_i = \frac{c}{2(1 + \sum_{j=1}^3 \alpha_j)^2}$$

where c is the circumference of the triangular facet F_i and α_j the smallest angle between two principal direction vectors computed at vertices of the facet.

Input: Vertex P_0 and its corresponding principal direction vector t_0
Principal direction vector t_i at each vertex P_i
Maximum length of the line of curvature max_length

Output: List of points L_k defining the line of curvature.

$L_0 = P_0$

$k = 0$

F_0 is the facet adjacent to P_0 which is pointed by t_0

while ($i < max_length$) AND (L_k not on a boundary edge of the 3D mesh)

do

begin

Let F_{k+1} the facet containing L_k and opposite to F_k .

Compute the principal direction vector T_k at point L_k by barycentric interpolation of the principal direction vectors t_i associated with the three vertices P_i of the facet F_{k+1} .

Project T_k on F_{k+1} and normalize the resulting vector. We get T_k^P .

Define the next point of the line of curvature by

$L_{k+1} = L_k + hT_k^P$, where h is a given value corresponding to the integration step.

if L_{k+1} is outside the facet F_{k+1} , compute the intersection point between L_kL_{k+1} and the traversed edge of F_{k+1} . L_{k+1} will then be relocated at this intersection.

$k = k + 1$

end

end

Algorithm 9: Computing lines of curvature on a 3D mesh.

This discrete integration scheme is improved in [ALL 03]. The authors compute T_k^P by using a 2D curvature tensor which is obtained by a discrete conformal parameterization which locally flattens the surface. They distinguish two sets of lines of curvature: one corresponds to the minimum principal curvatures and the other to the maximum principal curvatures. They also state that a line of curvature either starts from an umbilic and ends at another one, or is closed, or finishes on a mesh boundary. They first detect umbilics in order to have starting points to compute the lines of curvatures.

Then, they apply an accurate numerical scheme based on a fourth-order Runge–Kutta with an adaptive step based on a so-called “deviator” tensor.

In [MOR 10], the authors detail a method to detect the closeness of a line of curvature. It consists of monitoring the distances between \mathbf{P}_0 and the points of the line part L^+ and the ones of L^- . If the minimum distances to L^+ and L^- both become small, it means that L^+ and L^- go along the 3D mesh and then approach very close to \mathbf{P}_0 ; the line of curvature can then be considered as closed. Nevertheless, in most cases, the two parts L^+ and L^- are not really connected. Two signed distance fields d^+ and d^- are locally built on the 3D mesh from the points of L^+ and L^- . The sign is given by a projection on the principal direction vector orthogonal to the line of curvature, assuming that the frame of principal directions is direct. In fact, this separates locally the 3D mesh into a “left” side and a “right” side with respect to the line part. As the direction of integration of L^+ and L^- are opposite, the isoparametric line $d^+(\mathbf{P}) + d^-(\mathbf{P}) = 0$ corresponds to an average line between the two part lines. By construction, it is closed so it is a good approximation of the closed line of curvature L going through \mathbf{P}_0 .

In [KAL 09], the authors generalize the algorithm 9 in the case we have no more a 3D mesh but an unorganized 3D point cloud (a simpler method based on the same framework can be found in [PAN 10a]). They first compute the differential parameters with algorithm 8. Then they apply the same idea as in algorithm 9. Suppose that we begin from point \mathbf{P}_0 , we compute the principal direction vector \mathbf{t}_0 and we estimate the next point of the line of curvature by: $\mathbf{P}_1 = \mathbf{P}_0 + h\mathbf{t}_0$. Nevertheless, \mathbf{P}_1 lies on the estimated tangent plane of \mathbf{P}_0 but not on the surface sampled by the point cloud which is unknown. So, a projection step has to be added. For this, we take the points of the cloud which are in the neighborhood of \mathbf{P}_0 and we estimate the surface locally (see [KAL 09] for details). \mathbf{P}_1 is then projected on this surface, giving the next point \mathbf{P}'_1 of the line of curvature. Differential parameters at \mathbf{P}'_1 are then interpolated from the ones computed at points of the neighborhood of \mathbf{P}_0 . The process is iterated by $\mathbf{P}'_{i+1} = \text{projection}[\mathbf{P}'_i + h\mathbf{t}_i]$ and gives points on the line of curvature. Note that at each step i , the value h is adapted according to the potential error on the principal direction. For this, another point is computed as: $\mathbf{P}''_{i+1} = \text{projection}[\mathbf{P}'_i + h\mathbf{t}_{i+1}]$ (so with the principal direction estimated at \mathbf{P}'_{i+1}). The step size is then modified as: $h_{i+1} = h_i \sqrt{\tau/\Delta}$ where $\Delta = \|\mathbf{P}'_{i+1} - \mathbf{P}''_{i+1}\|/\|\mathbf{P}'_{i+1} - \mathbf{P}'_i\|$ where τ is a predefined tolerance value.

How to improve the computation of lines of curvature?

In all these methods, the main difficulty is to reduce the accumulation of local errors when building the line of curvature. These errors are due to:

- the inaccuracy of vertex position, especially in the case of a scanned object;
- the inaccuracy in the computation of differential parameters due to the discretization of the 3D mesh;
- the drift during the discrete integration scheme.

A solution is to get some information about the pattern of lines of curvature on the 3D mesh in order to infer some global constraints. In [SOT 08], the authors characterize the set of lines of curvatures over a surface by the list of umbilic points, a first list of lines of curvature (called *separatrices*) which terminate or converge to an umbilic and a second list of closed lines of curvatures (called *cycles*) with some of their properties. This characterization allows us to state some theorems about the stability of the pattern of lines of curvature with respect to some classes of perturbations of the surface. In [ZHA 09], the authors extract on different surfaces what they call a \mathcal{P} graph composed of the umbilics and the separatrices. They compute all the differential parameters on parametric surfaces defined by implicit equations (see [CHE 07] for formulas), which allows us to get exact values. Nevertheless, they emphasize the difficulty to isolate precisely the locations of umbilics which are, by definition, singular points on the surface.

Note that we can find some similarities in 3D flow visualization where we have to compute streamlines based on a local vector field. In this case, several techniques to extract topological features as singular points (sinks, sources, vortices) or limit cycles have been proposed (see, for example, [ARM 11]). But they are applied in a 3D isotropically discretized space, which allows us to make many computations with a higher accuracy than for a 3D mesh representing a surface where the sampling is locally 2D and irregular. Moreover, flows can be modeled by physical equations (as Navier–Stokes) and many assumptions can be made, in particular to characterize singular points or the local shape of streamlines. Nevertheless, some techniques as the detection of closed streamlines proposed in [WIS 06] could be interesting to analyze the pattern of lines of curvature.

If we are able to find some global information about the pattern of lines of curvature, we can use them to improve the computation of these lines. In [VEM 93a] (see also [VEM 93b]), the authors propose to use a reference continuous surface \mathcal{S} (e.g. a tube) gridded by its lines of curvature which are explicitly computed (e.g. the meridians and the parallels in the case of a tube). This surface \mathcal{S} is then deformed in order to fit the vertices of the 3D mesh. At each step of the deformation, any point \mathbf{M} of the surface \mathcal{S} is associated with a point \mathbf{P} of the 3D mesh. The tangent vectors of the isoparametric curves at point \mathbf{M} are then aligned with the principal directions computed at \mathbf{P} , which deforms the surface \mathcal{S} . At the end, not only the deformable surface \mathcal{S} is fitted to the mesh, but it is possible to emphasize the lines of curvatures of the mesh by tracing the isoparametric lines of \mathcal{S} . Then, by using a template grid of lines, it is possible to map a complete set of lines of curvature and to enforce the orthogonality of the principal directions. Unfortunately, results remain limited to quite simple and smooth scanned surfaces such as a light bulb. Moreover, this method can work only if there is no umbilic on the 3D mesh, otherwise we have to select a template with the appropriate pattern of lines of curvature around the umbilic(s) (for a classification of these patterns, see [SOT 08]).

Note that if we were able to accurately compute a discrete net of lines of curvature over the surface mesh, i.e. a net composed of polygonal approximation of lines of curvature, we could get the value of the principal curvatures at any point of the mesh with a bounded error [BAU 10].

1.4.3. *Crest/ridge lines*

A concept used in many applications

Crest or *ridge lines* are intuitively defined as topographic features, which describe the valley or the hill profiles of a landscape. Nevertheless, as emphasized in [KOE 93], the mathematical definition of these lines is not straightforward and it was heavily discussed in particular at the beginning of the 20th Century. Equations of crest/ridge lines are then proposed according to a natural frame where the z -axis is along the gravity direction and gives the elevation. For example, in [KWE 94], the authors define these lines as the contour lines “forming modified V’s pointing” upstream (for the valleys or ravines) or downstream (for the ridges) which correspond to local extrema of curvatures along a contour line (see also [HOS 92], section 7.4.1). But, as

explained in [BRU 96], if we want to generalize the concept of crest/ridge lines to a 3D surface, we must drop the notion of a predefined direction and work in a local frame which is intrinsic to the surface. Note that two applications of crest/ridge lines to analyze the topography of a landscape are presented in sections 3.4 and 3.5.

Crest/ridge lines can also be seen as the edges of a manufactured object. In particular, if this object is modeled as a 3D mesh by a computer-aided design procedure, we can just threshold the value of the dihedral angle between two contiguous facets and keep only the most salient edges. Nevertheless, when the 3D mesh is given by a 3D scan of the object, the potential low resolution and/or noise may result in unstable or inaccurate lines. We can then use other measurements as the defect angle (see section 1.3.5) or use a large set of facets and smooth the measurements (see, for example, [HUB 01] or [JIA 08]). But, to get a robust method for noisy 3D meshes, it appears that it is necessary to take into account principal curvatures and directions [PAG 02, VID 11] especially if we want to get continuous lines and not a set of disconnected sharp edges.

Crest/ridge lines are also anatomical terms. They define the prominent borders of some anatomical (sub)structures as, for example, the “cusp ridges” of a tooth, the “alveolar ridges” of the mouth, the “sagittal crest” of the skull or the “iliac crest” of the pelvis. We can also find equivalent terms as the “gyri”, which are the ridges on the cerebral cortex or the “orbital rim” which is the crest lines around the orbital opening. The huge development of 3D medical imaging systems allows radiologists to easily have access to 3D meshes of the anatomical structures of their patients [LEV 12]. Detecting automatically crest/ridge lines could assist medical doctors to localize and visualize precisely anatomical structures (see, for example, some applications to the brain in [STY 04], craniofacial anatomy in [ZHE 17] and paleo-anthropology in section 3.3), to compare them by registration (see, for example, applications to the brain in [SUB 99a], the skull in [GUÉ 94, REN 12] or the mandible in [AND 01]), to analyze their shape (e.g. to identify some anatomical facial features [KEN 96] or to make a diagnosis in craniofacial diseases [SUB 97]) or to plan a surgical procedure [SUB 98].

A survey of mathematical definitions

Around the middle of the 1990s, several applications to describe the shape of a 3D mesh based on *crest* or *ridge* lines were presented and there was a convergence of different mathematical definitions.

A first definition was proposed in [POR 71] (for a more detailed and extended version, see also [POR 01]) which is based on the focal surfaces (i.e. the sets of centers of principal curvatures; see section 1.2.9). Let us call E , the focal surface defined by one principal direction over the surface Σ . Now, if we follow the focal points $\mathbf{C}(\mathbf{P})$ corresponding to the points \mathbf{P} of a given line of curvature of Σ , we define on E a *focal curve*. If the focal curve fails to be regular at one point of E (i.e. its derivative vanishes, which means the curve stops and backtracks on itself forming a cusp), then the corresponding point on Σ is said to be a *ridge point*. As it can be shown that the set of non-regular points on E is a smooth curve, we can deduce that ridge points form ridge lines on Σ .

A second definition can be inferred from a classification of the focal points $\mathbf{C}(\mathbf{P})$ and was proposed in the first edition of [POR 01]. In [BRU 96] (see also [BRU 99] for details and [CAZ 05b] for an overview), we can find a sketch of the demonstration. Let us write a local approximation of the surface Σ at order 3 around the point \mathbf{P} , which extends the one described in section 1.2.8. We then get in a frame defined by the principal directions and the normal vector:

$$z(x, y) = \frac{1}{2}(k_1x^2 + k_2y^2) + b_0x^3 + b_1x^2y + b_2xy^2 + b_3y^3 + O(x^4, y^4) [1.35]$$

where k_1 and k_2 are the principal curvatures at point \mathbf{P} and $O(x^4, y^4)$ means terms in x or y with an order higher or equal to 4. In this frame, we can define the *focal spheres* S_1 and S_2 which are centered at focal points respectively located at positions $(0, 0, 1/k_1)$ and $(0, 0, 1/k_2)$ and which go through \mathbf{P} . Now, if we want to analyze the intersection between the focal spheres and the surface Σ , we have to solve (for the focal sphere S_2):

$$x^2 + y^2 + (z(x, y) - 1/k_2)^2 = 1/k_2^2$$

Bu using equation [1.35], we can write the equation of the projection in the plane (x, y) of the intersection curve between Σ and S_2 . If we limit to order 3, we get:

$$x^2 \left(1 - \frac{k_1}{k_2}\right) - \frac{2}{k_2}(b_0x^3 + b_1x^2y + b_2xy^2 + b_3y^3) + O(x^4, y^4) = 0$$

If \mathbf{P} is not an umbilic, we get $k_1 \neq k_2$ and we have a term in x^2 . Then, if $b_3 \neq 0$, the intersection curve can be approximated as something like $x^2 \pm y^3 = 0$ which is a single cusp. But if $b_3 = 0$, the intersection curve becomes something like $x^2 \pm y^4 = 0$ which is a tacnode or a double cusp. We can consider that, in this case, the focal sphere S_2 has a closer contact with the surface Σ and that \mathbf{P} is a *ridge point*.

In fact, this geometric definition can be transformed in a simple equation relative to k_2 and its associated principal direction \mathbf{t}_2 . For that (for some mathematical details, see [BEL 97, CAZ 05b]), we use again equation [1.35]. As the frame (x, y) is along the principal directions, the normal section $S_{\mathbf{t}_2}$ along \mathbf{t}_2 can be defined by:

$$s_{\mathbf{t}_2}(y) = \frac{1}{2}k_2y^2 + b_3y^3 + O(x^4, y^4)$$

We can compute the curvature of this 2D curve by derivation and if we approximate the formula to order 1 in y , this leads to an approximation of k_2 around \mathbf{P} along its associated principal direction \mathbf{t}_2 :

$$k_2(y) = k_2 + 6b_3y + O(y^2)$$

We have seen before that \mathbf{P} is defined as a ridge point when $b_3 = 0$. In this case, the above equation shows that k_2 reaches a local extremum at point \mathbf{P} along the principal direction \mathbf{t}_2 . We can do the same process for k_1 so we can write the following general definition for $j \in \{1, 2\}$

$$\mathbf{P} \text{ is a ridge point when } \nabla k_j(\mathbf{P}) \cdot \mathbf{t}_j(\mathbf{P}) = 0 \quad [1.36]$$

In [POR 01], it is shown that this definition of ridge points which is based on extrema of principal curvatures is equivalent to the first definition which was based on focal surfaces. This implies that the ridge points defined by equation [1.36] form *crest/ridge lines* on the surface Σ .

Equation [1.36] can also be found in [HOS 92] where it defines the *principal curvature extremum curves*.

Now, if we set $k_{amax} = \max(|k_1|, |k_2|)$ (note that we use absolute values) and \mathbf{t}_{amax} its associated principal direction. The equation:

$$\nabla k_{amax}(\mathbf{P}) \cdot \mathbf{t}_{amax}(\mathbf{P}) = 0 \quad [1.37]$$

defines a subset of ridge points called *crest points* in [BRU 96]. These points form the *crest lines* which are then a subset of the ridge lines. In fact, we can find a similar definition in a previous paper [GUÉ 93] even though the definition of k_{amax} was not clearly presented. In this chapter, it is shown that crest lines characterize the salient curves on a surface.

Note that the lines, defined by the equation $\nabla k_i(\mathbf{P}) \cdot \mathbf{t}_j(\mathbf{P}) = 0$ where $j \neq i$ (that is k_1 reaches an extremum along the principal direction \mathbf{t}_2 or k_2 reaches an extremum along the principal direction \mathbf{t}_1) are called *sub-parabolic lines* (as there is a connection with the parabolic points (see section 1.2.8) of the focal surface) and are thoroughly analyzed in [MOR 96].

In [BEL 96], the authors propose five different definitions of ridge lines based on extrema of curvatures. These extrema are not only along the associated principal direction but may also be along the normal or the vertical section curve passing through \mathbf{P} in the direction of the maximum principal curvature. The authors then show some connections between all these different definitions.

A third definition is based on the concept of symmetry. In [NAC 85], the Symmetric Axis Transform is generalized to 3D surface by using spheres which touch the surface at two points; the symmetric axis then becomes a symmetric surface. In [YUI 90], the authors prove that the intersection of these symmetry surfaces with the surface are generated by and only by lines on the surface corresponding to the extrema of principal curvature. Nevertheless, they do not use explicitly the term crest or ridge line. Moreover, the authors of [ANO 94a] emphasize that the symmetric axis transform can be used only to find the ridge lines, which are along the convexities of the surface and not those which are along the concavities. They propose then in [ANO 94b] a new definition of the skeletonization based on singularity theory in order to define all ridge lines. In this chapter, they also make a

connection between the definitions based on the focal surfaces, the differential singularities and symmetry (see also [BEL 96]).

A fourth definition is based on an implicit definition of the surface in a 3D image. Let a 3D image (i.e. a regular grid) I where at each element (called a *voxel*) of coordinates (x, y, z) is associated a scalar value $I(x, y, z)$. As, for a 2D image, we can compute the differentials of intensity, at any order, along the three axes (as, for example, $\frac{\partial I}{\partial x}$ or $\frac{\partial I^2}{\partial x \partial y}$). The implicit equation $I(x, y, z) = I_0$ defines then a surface Σ called an *isosurface* of the 3D image I . In [THI 95], we can find the formulas to compute differential parameters at any point $\mathbf{P}(x, y, z)$ of Σ based on the intensity and its derivatives at the corresponding voxel. For example, the first coefficient of the first fundamental form is given by:

$$E(x, y, z) = \left(\frac{\partial I(x, y, z)}{\partial x}^2 + \frac{\partial I(x, y, z)}{\partial z}^2 \right) / \frac{\partial I(x, y, z)}{\partial z}^2$$

Note that [SAN 90] and [MON 95] also propose some formulas based on image differentials, but they are based on a local parametric representation of the isosurface (obtained by fitting a surface patch in the case of the first reference) which is less general than using the implicit representation.

In [MON 92] (for a more detailed version, see also [MON 95]), the authors use such formulas to detect what they called the ridge and valley points on Σ . These points correspond to the zero-crossing of what they call the *extremality criterion* e which is defined as:

$$e = \nabla k_{max} \cdot \mathbf{t}_{max} = 0$$

In fact, this is exactly the same that equation [1.37]. In [THI 96a], this extremality criterion is generalized to the two principal curvatures and their associated principal directions resulting in four different extremality zero-crossings which define *extremal lines* (which include the ridge lines). In [EBE 94], the authors extend the definition of the ridge and valley points to n -dimensional images by generalizing the concept of isosurface to a specific level set of dimension $n - 1$. They also point the connection between this image-based definition and the symmetry-based definition.

In the following, we will call *crest/ridge function* the generalized version of equation [1.36]:

$$e_j(\mathbf{P}) = \nabla k_j(\mathbf{P}) \cdot \mathbf{t}_j(\mathbf{P}) \quad [1.38]$$

where j may be either one of the two principal curvatures (and associated principal directions) ($j \in \{1, 2\}$) or the maximum ($j = \max$) or minimum ($j = \min$) one or the maximum in absolute value ($j = \max$). The equation $e_j = 0$ defines then crest/ridge points and crest/ridge lines.

Computation of crest/ridge lines on parametric surfaces

If we assume that we have a parametric form $f(u, v)$ of the surface Σ , we are able to compute formally, and then exactly, without any numerical approximation, all the differential parameters. Nevertheless, some problems must be investigated before proposing a method to compute the crest/ridge lines.

At any point \mathbf{P} except umbilics, we have two principal directions that we can order with respect to the value of their associated principal curvatures. We then define $k_{\min} = \min(k_1, k_2)$ and $k_{\max} = \max(k_1, k_2)$ which are respectively associated with their corresponding principal directions \mathbf{t}_{\min} and \mathbf{t}_{\max} . For clarity in the following, we will call, according to [POR 01], *blue* crest/ridge lines the lines defined by the equation $e_{\max} = \nabla k_{\max}(\mathbf{P}) \cdot \mathbf{t}_{\max}(\mathbf{P}) = 0$ and *red* crest/ridge lines those defined by the equation $e_{\min} = \nabla k_{\min}(\mathbf{P}) \cdot \mathbf{t}_{\min}(\mathbf{P}) = 0$. Note that some researchers as [BRU 96] or [THI 96a] order the principal directions according to their absolute values. Blue and red ridge lines may then change of colors but keeps the same structure.

Blue and red ridge lines do not auto-intersect but may cross in so called *purple points* which were first described in [POR 01] and studied in [CAZ 06]. Umbilic points can also be considered as crest/ridge points since they are in the closure of crest/ridge lines. More precisely, it can be shown (see, in particular, [CAZ 05b]) that generic umbilics are of two types: either three crest/ridge lines or only one crest/ridge line cross at an umbilic. This property allows us to define a graph of crest/ridge lines all over a closed surface where the nodes are umbilics or purple points (see, in particular, the ellipsoid example in [CAZ 05b]). Such a graph description was also proposed under the name of the *extremal mesh* in [THI 96a].

A crucial problem is that the principal directions are not intrinsically oriented along their corresponding line of curvature (see section 1.2.3). If we invert the orientation of the principal direction frame $(\mathbf{t}_1, \mathbf{t}_2)$ in $(-\mathbf{t}_1, -\mathbf{t}_2)$, e_{min} and e_{max} turn respectively into $-e_{min}$ and $-e_{max}$. But then, how to compute the zero-crossings of an expression whose sign is not clearly defined?

One solution is to impose a continuity of the orientation of the principal direction \mathbf{t}_i in a neighborhood which can be defined in space or along the corresponding line of curvature. This means that if \mathbf{P} and \mathbf{P}' are close, the two principal directions make an acute angle, i.e. $\mathbf{t}_i(\mathbf{P}) \cdot \mathbf{t}_i'(\mathbf{P}') > 0$. This constraint is described in [MOR 96] and called the *Acute Rule* in [CAZ 06].

Another solution is to use the Gaussian extremality [THI 96a] which is defined by: $e_g(\mathbf{P}) = e_{min}(\mathbf{P}) \cdot e_{max}(\mathbf{P})$. In this case, if we assume that the frame $(\mathbf{t}_{min}, \mathbf{t}_{max})$ is direct, changing the orientation of \mathbf{t}_{min} results in changing the orientation of \mathbf{t}_{max} and that does not change the sign of e_g . But then, we get all the crest/ridge lines without being able to distinguish between the blue and red ones.

We can classify the methods to compute crest/ridge lines on parametric surfaces into five categories: formal computation of zero-sets, numerical computation of zero-sets, incremental tracing, marching by interpolation or integration of a direction field.

In the first category, we formally solve the crest/ridge equation $e_i = 0$ by computing explicitly the formulas of partial derivatives to order 3 of $f(u, v)$. But this is computationally very expensive; for example, it is shown in [CAZ 08a] that in the case of a Bézier surface of degree 4, it requires to solve a bivariate polynomial of total degree 84, with 1,907 terms! This makes methods of this category impossible to use on complex parametric surfaces as NURBS.

In the second category of methods, we numerically evaluate the crest/ridge function e_i at only some points of the surface. A first method was described in [HOS 92], section 7.4, where it is proposed to find the extremum points of curvature along lines of curvature. As it is too complex to formally solve the mathematical equations, the idea is to determine these extremum points numerically by interpolating points which are sampled along the line of

curvature. Nevertheless, in this chapter, the computed points are not linked to form crest/ridges lines. In [GUÉ 93, GUÉ 95], the parametric surface is defined by a tensor product of spline functions. By sampling regularly the parameter space it becomes possible to compute the value of the curvature k_i on the surface and detect the local maximum points. These points are then linked based on their proximity. The author compares this method with the numerical resolution of the crest/ridge line equation $e_i = 0$ and concludes that results are similar.

The third category of methods proposes to trace incrementally small parts of the crest/ridge lines. For example, the algorithm described in [MUS 11] is based on the property that crest/ridge lines intersect transversely the corresponding curvature lines (i.e. the ones which are defined with the same differential parameters k_i and t_i when compared to the crest lines) except at a few isolated points (called *turning points*) where they are tangent.

Let us assume that we want to compute the crest/ridge lines based on $e_1 = 0$. The idea is to begin at a seed crest/ridge point \mathbf{R} . Then we are going to take a neighbor point \mathbf{P} on the surface. Then, if we slide \mathbf{P} along the curvature line defined by t_1 , it will cross the crest/ridge line. Once we have determined the intersection point, we can link it to \mathbf{R} in order to get a local linear approximation of the crest/ridge line. Starting from \mathbf{R} , the algorithm is then composed of the following steps:

- define a neighbor point on the surface \mathbf{R}_{adv} . This is practically performed by advancing of a step ϵ_{adv} along the line of curvature defined by $t_2(\mathbf{R})$ (so orthogonally to $t_1(\mathbf{R})$) and by projecting the point $\mathbf{R} + \epsilon_{adv}t_2(\mathbf{R})$ on the surface;
- now, define the point \mathbf{R}_{slide} on the surface by sliding of a step ϵ_{slide} from \mathbf{R}_{adv} along the line of curvature defined by $t_1(\mathbf{R}_{adv})$ and projecting the point $\mathbf{R}_{adv} + \epsilon_{slide}t_1(\mathbf{R}_{adv})$ on the surface;
- test whether \mathbf{R}_{slide} is a crest/ridge point using the equation $e_1(\mathbf{R}_{slide}) = 0$:
 - if not, iterate again the sliding step from \mathbf{R}_{slide} ,
 - if a crest/ridge point is reached, use \mathbf{R}_{slide} as a starting point to continue the tracing of the crest/ridge line and go back to step 1 of the algorithm.

Note that the Acute Rule is applied to orient consistently principal directions. The set of seed crest/ridge points is defined by finding the critical points of curvature (i.e. where the curvature gradient is equal to zero) and the umbilics. In fact, umbilics are included as seed points since crest/ridge lines may pass through these points. This algorithm has been successfully tested on rational tensor product B-spline surface representations.

An example of the fourth category of methods is given in [CLÉ 08]. This is called a marching method as it is based on a regular discretization of the parameter space and a tracing by interpolation from one discrete element to a neighbor one. More precisely, we can use a regular grid where each node (u_j, v_k) corresponds to a point $\mathbf{P}_{j,k}$ on the parametric surface. We compute then the crest/ridge function at this node by $e_i(j, k) = e_i(\mathbf{P}_{j,k})$. Now, for each square of the grid which is defined by four nodes, if at least one node has a negative crest/ridge value and at least one node has a positive crest/ridge value, we can assume that a line segment corresponding to $e_i = 0$ crosses the square and we can draw it based on interpolation of values $e_i(j, k)$ of the square. At the end, we link all the line segments to get a continuous line going through the parameter space. This line corresponds in the 3D space to a crest/ridge line. Of course, the geometric accuracy is directly related to the discretization step. This implies that it may be difficult to detect some intersection, loop or tangency problems if the step is not small enough.

Note that the “Marching Lines” term is also used in [THI 96b]. In this case, the parametric surface is defined by an implicit equation. If we discretize the space into a 3D regular grid of cubes (also called voxels), we can approximate each intersection between the surface and a cube by a patch composed of one or several polygons. If we compute now, at each vertex of the patch, the crest/ridge function, we can interpolate line segments which will form crest/ridge lines.

The fifth category of methods is based on the construction of a vector field in the parameter space. Our goal is to compute the crest/ridge lines which correspond to the isolines $e(u, v) = 0$. By definition, when a point $\mathbf{P}(u, v)$ moves along this isoline, we have $\frac{\partial e}{\partial u} du + \frac{\partial e}{\partial v} dv = 0$. This implies that the tangent vectors of isolines form in the parameter space a vector field defined by: $(\frac{\partial e}{\partial v}, -\frac{\partial e}{\partial u})$.

Now, if we start from a crest/ridge point $\mathbf{P}_0(u_0, v_0)$, we can integrate from (u_0, v_0) the vector field and we build a line in the parameter space which corresponds to a crest/ridge line in the 3D space.

In [CLÉ 08], we find an example of an implementation of such an integration method. The algorithm is composed of the following steps:

- a uniform random distribution of points is generated all over the parameter space. We select the points with a small value of the crest/ridge function. Then by applying a conjugate gradient-based algorithm, we can find the local minima of this function. This will define crest/ridge points (including umbilics) \mathbf{P}_i ;

- numerical instability often appears when computing the partial derivatives of e around the points \mathbf{P}_i . So, we define a circle centered in \mathbf{P}_i with a radius so that we are far from the instability area. On this circle, we can find one or several points \mathbf{Q}_i^k which have small values of the crest/ridge function. The set of \mathbf{Q}_i^k and the vectors $\mathbf{P}_i\mathbf{Q}_i^k$ will be respectively the initial points and the initial directions for the integration process starting from \mathbf{P}_i ;

- as there may be some error accumulation during integration, the crest/ridge line starting from a point \mathbf{Q}_i^k (which is linked to the point \mathbf{P}_i) may diverge and not reach another crest/ridge point \mathbf{P}_j . The idea is then to build a Voronoï tessellation of the parameter space based on the set of all \mathbf{Q}_i^k (then for all i and k). We will integrate the crest/ridge line inside each Voronoï cell until it reaches its border. Then we are sure to have line parts which are linked to points \mathbf{P}_i . A fifth-order Runge–Kutta method is used for numerical integration;

- now, when the extremities of two line parts coming from \mathbf{Q}_i^k and \mathbf{Q}_j^l are close on both side of a Voronoï cell, we connect them and we get a longer crest/ridge line part which links \mathbf{P}_i and \mathbf{P}_j . We update the Voronoï tessellation after removing \mathbf{Q}_i^k and \mathbf{Q}_j^l , we apply the integration process and link another couple of points if it possible.

The authors tested the method on a fourth-order polynomial function and emphasize that its efficiency is strongly related to the accuracy of the numerical integration scheme.

Another integration method is described in [CHE 11]. It is also based on a vector field computed in the parameter space. In order to deal with the problem of error accumulation, the authors propose to project the estimation

of a new crest/ridge point on its corresponding line of curvature in order to find the optimal integration step. This method can deal with some singular points as turning points and was tested on several bicubic B ezier patches which correspond, in particular, to manufactured objects.

Computation of crest/ridge lines on a discrete 3D mesh

One of the first methods to compute crest/ridge lines on a discrete 3D mesh was proposed in [WAT 01]. In fact, it does not belong to any category of methods seen in the previous section. This method is based on the detection of particular configurations of the focal surfaces by using the following property demonstrated in [LUK 98] and [WEA 55], item 75.

Let us work in the frame of the lines of curvature $(\mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$. We assume that none of the principal curvatures is null which allows us to define the focal surfaces E_1 and E_2 (see section 1.2.9). If \mathbf{P} moves a little over Σ , it will sweep the area:

$$A = \left\| \frac{\partial \mathbf{P}}{\partial u} \times \frac{\partial \mathbf{P}}{\partial v} \right\|$$

At the same time, the corresponding focal point \mathbf{C}_1 will move over E_1 and sweep an area given by:

$$A' = \left\| \frac{\partial \mathbf{C}_1}{\partial u} \times \frac{\partial \mathbf{C}_1}{\partial v} \right\|$$

We have seen at the end of section 1.2.9 that for a point \mathbf{P} :

$$\begin{aligned} \frac{\partial \mathbf{C}_1}{\partial u} &= \frac{1}{k_1^2} \frac{\partial k_1}{\partial u} \mathbf{n} \\ \frac{\partial \mathbf{C}_1}{\partial v} &= \left(1 - \frac{k_2}{k_1}\right) \frac{\partial \mathbf{P}}{\partial v} + \frac{1}{k_1^2} \frac{\partial k_1}{\partial v} \mathbf{n} \end{aligned}$$

We then get:

$$A' = \left| \left(1 - \frac{k_2}{k_1}\right) \frac{1}{k_1^2} \frac{\partial k_1}{\partial u} \right| \left\| \frac{\partial \mathbf{P}}{\partial v} \right\|$$

which allows us to compute the area-ratio r_1 :

$$r_1 = \frac{A'}{A} = \frac{\left| \left(1 - \frac{k_2}{k_1}\right) \frac{1}{k_1^2} \frac{\partial k_1}{\partial u} \right| \left\| \frac{\partial \mathbf{P}}{\partial v} \right\|}{\left\| \frac{\partial \mathbf{P}}{\partial u} \right\| \left\| \frac{\partial \mathbf{P}}{\partial v} \right\|}$$

$$r_1 = \frac{\left| \left(1 - \frac{k_2}{k_1}\right) \frac{1}{k_1^2} \frac{\partial k_1}{\partial u} \right|}{\left\| \frac{\partial \mathbf{P}}{\partial u} \right\|}$$

This means that r_1 tends to 0 if and only if:

– $k_1 = k_2$ which means that the point \mathbf{P} is an umbilic.

– $\frac{\partial \mathbf{P}}{\partial u} = 0$ when \mathbf{P} moves along the line of curvature defined by \mathbf{t}_1 .

This corresponds exactly to the definition of the crest/ridge point given by equation [1.36].

Of course, we can do exactly the same for the other focal point \mathbf{C}_2 moving on E_2 and we get an area-ratio r_2 with the same property along the line of curvature defined by \mathbf{t}_2 .

The authors of [WAT 01] apply then this “area degenerating” property in the discrete case of a 3D mesh. Note that they explain that they use this property instead of directly detecting non-regular points on a focal curve (see the first definition in the section *Survey of mathematical definitions* of section 1.4.3) as this latter method is too unstable for complex 3D meshes.

Let us assume that we have computed the normal vector $\mathbf{n}(\mathbf{P}_i)$ and the principal curvatures $k_1(\mathbf{P}_i)$ and $k_2(\mathbf{P}_i)$ on all vertices of the 3D mesh \mathcal{M} by any of the method described in section 1.3. For each facet F_i of the 3D mesh (which is assumed triangular), we can define the three focal points corresponding to the three vertices of the face, $\mathbf{P}_i^0, \mathbf{P}_i^1, \mathbf{P}_i^2$:

$$\mathbf{C}_i^0 = \mathbf{P}_i^0 + \frac{1}{k_j(\mathbf{P}_i^0)} \mathbf{n}(\mathbf{P}_i^0)$$

$$\mathbf{C}_i^1 = \mathbf{P}_i^1 + \frac{1}{k_j(\mathbf{P}_i^1)} \mathbf{n}(\mathbf{P}_i^1) \quad \mathbf{C}_i^2 = \mathbf{P}_i^2 + \frac{1}{k_j(\mathbf{P}_i^2)} \mathbf{n}(\mathbf{P}_i^2)$$

where k_j corresponds either to k_1 or k_2 .

We can compute then two area-ratios r_1 and r_2 for the facet F by:

$$r_j(F) = \frac{A(\mathbf{C}_i^0 \mathbf{C}_i^1 \mathbf{C}_i^2)}{A(\mathbf{P}_i^0 \mathbf{P}_i^1 \mathbf{P}_i^2)}$$

where $j \in \{1, 2\}$ depending if we choose k_1 or k_2 .

According to the above property, a facet F where one of the area-ratio $r_1(F)$ or $r_2(F)$ is high (a threshold value 30% of all the area-ratios of the mesh facets is proposed in the paper) contains potentially a crest/ridge point. Note that to have good results on complex meshes, it is necessary to smooth the normal vector coordinates and curvature values at a vertex by a nonlinear averaging in its neighborhood.

Nevertheless, a crest/ridge facet with a small area-ratio may be, in fact, a flat umbilic. If we compute for this facet F a *curvedness* value $c(\mathbf{P}_i^j) = |k_1(\mathbf{P}_i^j)| + |k_2(\mathbf{P}_i^j)|$ at its three vertices and apply a threshold (as, for example, the 50% percentile of all the values computed on the entire 3D mesh), we can discard the potential crest/ridge facets which are not enough salient.

Nevertheless, we have a set of crest/ridge facets but not lines. For this, the authors propose to use a thinning technique which generalizes the skeletonization process, well-known in image processing. The idea is to “peel-off” the set of crest/ridge facets in order to keep only a one-triangle width undisconnected band of facets (for more details, see the next section).

In [YU 08], the authors have developed both CPU and GPU-based algorithms in order to efficiently approximate the focal meshes. They could then be used to apply the above method to very large 3D meshes.

All the other methods which have been proposed to compute crest/ridge lines on a discrete 3D mesh belong only to the category “numerical computation of zero-sets” defined in the previous section. In the following, we describe some of the most referenced methods.

In the method proposed in [STY 04], the first step consists of detecting the crest/ridge points on the 3D mesh and is described in the following algorithm 10:

Input: 3D mesh composed of vertices \mathbf{P}_i
Output: Flag $f_i \in \{false, true\}$ ($f_i = true$ if \mathbf{P}_i is a crest/ridge point)
begin
 forall vertex \mathbf{P}_i **do**
 Compute the differential parameters by fitting locally a quadric to the mesh (see section 1.3.4).
 We have then the principal curvatures (k_1^i, k_2^i) and the associated principal directions $(\mathbf{t}_1^i, \mathbf{t}_2^i)$.
 end
 forall vertex \mathbf{P}_i **do**
 Select the maximum principal curvature $k_{max} = \max(k_1^i, k_2^i)$ and the associated principal direction \mathbf{t}_{max} .
 Project \mathbf{t}_{max} on the facets connected to \mathbf{P}_i .
 Find the two facets F_i' and F_i'' which contain the projection of \mathbf{t}_{max} . These two facets are on opposite sides with respect to \mathbf{P}_i .
 The projection of \mathbf{t}_{max} intersects the edges of F_i' and F_i'' , opposite to \mathbf{P}_i , respectively at the points \mathbf{P}_i' and \mathbf{P}_i'' .
 By linear interpolation of the maximum curvature of the two vertices of the edge, it is possible to compute the maximum principal curvature k_{max}' at \mathbf{P}_i' and k_{max}'' at \mathbf{P}_i'' .
 If $((k_{max})^2 - (k_{max}')^2) > \epsilon$ AND $((k_{max})^2 - (k_{max}'')^2) > \epsilon$, this means that the maximum curvature of \mathbf{P}_i is locally maximum in its corresponding principal direction, i.e. \mathbf{P}_i is a crest/ridge point. In this case $f_i = true$ else $f_i = false$.
 end
end

Algorithm 10: Detect crest/ridge points in a 3D mesh by the method [STY 04].

The problem is then to link crest/ridge points in order to build lines over the 3D mesh. First, a region growing process is applied in order to create for each crest/ridge point, a crest/ridge region corresponding to its 2-neighborhood (i.e. the points which belong to the neighborhood of the neighborhood). Then, as for the focal method described before, we use a skeletonization process in order to obtain a set of connected lines along the edges of the 3D mesh.

Another method was proposed in [OHT 04] and is described in the following algorithm 11:

Input: 3D mesh composed of vertices \mathbf{P}_i

Output: Crest/ridge lines as a set of segments

begin

forall *vertex* \mathbf{P}_i **do**

Construct an implicit surface which approximates locally the vertices of the neighborhood $N(\mathbf{P}_i)$ as their normal vectors. Based on this parametric representation, it is possible to compute the principal curvatures (k_1^i, k_2^i) and the associated principal directions $(\mathbf{t}_1^i, \mathbf{t}_2^i)$ at vertex \mathbf{P}_i (see the end of section 1.2.5). Select the maximum principal curvature $k_{max}^i = \max(k_1^i, k_2^i)$ and the associated principal direction \mathbf{t}_{max}^i . As we have a parametric representation, we can compute the crest/ridge function $e_{max}^i = \nabla k_{max}^i \cdot \mathbf{t}_{max}^i$ at the vertex \mathbf{P}_i (see equation [1.38]).

end

forall *edge* $\mathbf{P}_i\mathbf{P}_j$ **do**

If the angle between \mathbf{t}_{max}^i and \mathbf{t}_{max}^j is obtuse, flip \mathbf{t}_{max}^i which inverts the sign of e_{max}^i . This corresponds to the Acute Rule constraint (see section 1.4.3).

If $e_{max}^i \cdot e_{max}^j < 0$ then there is a zero-crossing of e_{max} which corresponds to a crest/ridge point on the edge.

If we want to recover only the maxima of k_{max} , this means that when you go along the edge from \mathbf{P}_i to \mathbf{P}_j , the derivative of k_{max} along the direction \mathbf{t}_{max} should be positive, then negative. This can be approximated by one of the conditions:

$$(\mathbf{P}_i\mathbf{P}_j \cdot \mathbf{t}_{max}^i) e_{max}^i > 0$$

$$(\mathbf{P}_j\mathbf{P}_i \cdot \mathbf{t}_{max}^j) e_{max}^j > 0$$

If one condition is true, approximate the position of the crest/ridge point \mathbf{P}_{ij} by linear interpolation of the positions of \mathbf{P}_i and \mathbf{P}_j , respectively weighted by e_{max}^i and e_{max}^j .

end

forall *facet of the mesh* **do**

If 2 crest/ridge points belong to the edges of the facet (which is assumed triangular), connect them by a segment.

If all the three edges of the facet contain crest/ridge points, compute the centroid of these three points, connect it to the three points by three segments.

end

end

Algorithm 11: Extract crest/ridge lines on a 3D mesh by the method [OHT 04].

In [KIM 05], the method to connect crest/ridges points is modified in order to make it more robust. There is no more linear interpolation: either the vertex \mathbf{P}_i or \mathbf{P}_j is selected as the crest/ridge point depending on the maximum value of k_{max} . Then, for each crest/ridge vertex \mathbf{P}_k , we examine all the neighboring vertices. If two or more crest/ridge vertices exist, we select the vertex \mathbf{P}_l such as the angle $\angle(\mathbf{P}_k\mathbf{P}_l, \mathbf{t}_{max}^k)$ is minimum. Note that the authors use a modified Moving Least-Squares approximation technique in order to reduce the computation time of the differential parameters.

In [HIL 05], the authors describe two methods to add a smoothing step in the algorithm 11 to get more visually pleasing crest/ridge lines.

In the first method, values of the crest/ridge point function e_{max}^i are regularized before approximating the crest/ridge point \mathbf{P}_{ij} , by applying a Laplacian smoothing (i.e. the value at a vertex is modified with respect to the values at the neighbor vertices). But the difficulty is to be independent of the sign of e_{max}^i which can be inverted, just by flipping the corresponding \mathbf{t}_{max}^i vector. The modified Laplacian smoothing is then defined by:

$$e_{max}^i \leftarrow e_{max}^i + \sum_{\mathbf{P}_j \in N(\mathbf{P}_i)} (w_{ij} e_{max}^j - e_{max}^i)$$

where $w_{ij} = \text{sgn}(\mathbf{t}_{max}^i \cdot \mathbf{t}_{max}^j)$, which corresponds in fact to the Acute Rule.

The second method consists of smoothing the crest/ridge lines after their extraction. We can use any smoothing scheme as, for example, the Laplace one (see, for example, [JIN 06]). The problem is then to ensure that the smoothed line does not deviate too much from the original one. The idea is to introduce a constraint based on the Hausdorff distance w.r.t. to the original line.

In [YOS 08], the authors propose to add to the algorithm 11 a step to reduce the fragmentation of the crest/ridge lines. The idea is to look in the neighborhood of each crest/ridge line endpoint. If we can find another endpoint belonging to another crest/ridge line, we connect both of them if the angle between their end segments are relatively small. An implementation of their algorithm which includes the estimation of differential parameters and

the tracing of crest/ridge lines is publicly available as C++ code associated with a Java3D viewer¹⁰.

Another method to compute crest/ridge lines is proposed in [CAZ 05d] and is publicly available in the open-source C++ Computational Geometry Algorithms Library CGAL¹¹. The algorithm is based on the topological analysis of the crest/ridge lines network described in [CAZ 06]. In particular, in [CAZ 05c], it is emphasized that only either 3 or 1 crest/ridge lines can cross a generic umbilic point. It becomes then possible, by assuming some general hypotheses about the triangulation of the 3D discrete mesh and by detecting umbilic points, to extract the crest/ridge lines in a topologically consistent way. This method can be decomposed into the following steps.

The first step consists of computing the differential parameters for each vertex of the 3D mesh. Even though the proposed method is the one described in [CAZ 05a], we could use any of the algorithms described in section 1.3. The second step aims to detect umbilic areas over the 3D mesh. The idea is to define the patch of neighborhood facets over the considered face. A vertex \mathbf{P} of this patch is an umbilic if $k_1(\mathbf{P}) = k_2(\mathbf{P})$ (see section 1.2.2). As we have differential parameters at some discretized vertices, we never have a strict equality and it requires to introduce a threshold ϵ to decide if \mathbf{P} is a potential umbilic vertex by $|k_1(\mathbf{P}) - k_2(\mathbf{P})| < \epsilon$. Nevertheless, as it is difficult to locate precisely an umbilic which is an isolated singularity point, it is better to detect potential umbilic facets which are the facets where all vertices are potential umbilic vertices. But, this may be not sufficient to detect umbilic areas with a good level of reliability. In [SAN 92], the authors propose to analyze locally the field of principal directions by using the *index function* defined at a point \mathbf{P} by:

$$index(\mathbf{P}) = \frac{1}{2\pi} \int_0^{2\pi} \langle \mathbf{t}_1(\mathbf{M}(\theta)), \mathbf{u} \rangle d\theta$$

where \mathbf{u} is a given direction vector, $\mathbf{M}(\theta)$ is a point, parameterized by the angle θ with respect to \mathbf{u} , which follows a small closed counterclockwise trajectory around the point \mathbf{P} . The index function is then the integral all around \mathbf{P} of the angle between a principal direction and a given direction. In fact, this function describes the way the lines of curvature turn around the

¹⁰ <http://www.riken.jp/briect/Yoshizawa/Research/Crest.html>.

¹¹ http://doc.cgal.org/latest/Ridges_3/index.html.

point P , and it is proven (see, for example, [CAZ 05c]) that for an umbilic point, the index is either $+\frac{1}{2}$ (the umbilic is then classified as *lemon* or *monstar*) or $-\frac{1}{2}$ (the umbilic is classified as *star*). Then, if we think that the considered facet is a potential umbilic facet, we approximate its index function by moving M around the boundary of the patch of its neighborhood facets. We confirm that this facet is an umbilic one if its index is equal $\pm\frac{1}{2}$ and the neighborhood patch of this umbilic facet becomes an umbilic area. More details about the index function and its computation around a vertex of a 3D discrete mesh can be found in [PAT 10], p. 236.

The authors propose then to compute some third order differential parameters to define if 1 or 3 crest/ridge lines go through the umbilic area even though there is no detail about how to approximate these parameters on a 3D discrete mesh.

The third step focuses on finding crest/ridge points on edges outside the umbilic areas. It uses the same idea than the previous algorithm 11: after applying the Acute Rule constraint, test if the crest ridge function has opposite sign at the two sides of an edge. If this is the case, compute the position of the crest/ridge point by linear interpolation. In a triangular facet, two crest/ridge points on consecutive edges are linked to form a crest/ridge segment.

The fourth step consists of tagging the crest/ridge segment as elliptic or hyperbolic. In fact, an elliptic crest/ridge point corresponds to a maximum of k_1 or a minimum of k_2 . This is equivalent to the detection of the maxima of k_{max} in the second loop of the previous algorithm 11 but the authors propose to use not only the two signs of the crest/ridge function on an edge but rather the three signs computed on the vertices of the facet crossed by the segment. This prevents from some ambiguities when an edge is almost parallel to a crest/ridge line. At the end, we get a set of tagged crest/ridge segment.

In the last step, by linking all the crest/ridge segments, we get crest/ridge lines. When these lines end at the boundary of an umbilic area, they are connected to the center of the umbilic area in order that they do not cross each other. According to theoretical topological constraints, there must be only one or three connections to perform but it may be more complex in the case of a discrete 3D mesh.

All these steps lead to the following algorithm 12.

AQ2

Input: 3D mesh composed of vertices \mathbf{P}_i which is assumed compliant w.r.t. to some general hypotheses

Output: Crest/ridge lines as a set of segments

begin

forall vertex \mathbf{P}_i **do**

 Compute the principal curvatures $k_1(\mathbf{P}_i), k_2(\mathbf{P}_i)$ and the associated principal directions $\mathbf{t}_1(\mathbf{P}_i), \mathbf{t}_2(\mathbf{P}_i)$.

end

forall facet F_i of the mesh **do**

 Take iteratively the neighbor facets which centers are within a given radius around the centroid of F_i . They define a patch \mathcal{P}_i of facets which has the topology of a disk around F_i .

forall facet F_i^j of the patch \mathcal{P}_i **do**

 Compute $\Delta k(F_i^j)$ as the arithmetic average of the values $k_1(\mathbf{P}) - k_2(\mathbf{P})$ computed at each vertex \mathbf{P} of the facet F_i^j .

end

if $F_i = \operatorname{argmin}_{F_i^j} \Delta k(F_i^j)$

then

 Apply the Acute Rule to orient consistently the principal directions of the vertices located on the boundary of the patch \mathcal{P}_i .

 By adding the angle deviation between one of the principal direction w.r.t. and a fixed direction, for all the vertices \mathbf{P}_i on the boundary of the patch \mathcal{P}_i , estimate the index function $\operatorname{index}(F_i)$.

if $\operatorname{index}(F_i) = \pm \frac{1}{2}$

then

 By computing some third order differential parameters, detect if 1 or 3 crest/ridge lines cross the facet F_i .

 Tag all the facets of the patch \mathcal{P}_i as “umbilic” facets.

end

end

end

forall edge $\mathbf{P}_i\mathbf{P}_j$ which does not belong to facet tagged “umbilic” **do**

 Compute the crest/ridge function e_i and e_j respectively at \mathbf{P}_i and \mathbf{P}_j .

 If $e^i \cdot e^j < 0$ then there is a zero-crossing of the crest/ridge function which corresponds to a crest/ridge point on the edge.

 Approximate the position of the crest/ridge point \mathbf{P}_{ij} by linear interpolation of the positions of \mathbf{P}_i and \mathbf{P}_j weighted by e^i and e^j .

end

forall facet F_i of the mesh **do**

 If 2 crest/ridge points belong to two consecutive edges of F_i (which is assumed triangular), connect them by a straight crest/ridge segment.

end

forall crest/ridge segment s **do**

 Find the facet F_s which contains the segment s .

 Tag the segment s as “elliptic” or “hyperbolic” according to the signs of the crest/ridge function at the vertices of F_s .

end

Link all the consecutive segments to form crest/ridge lines.

When these lines end at the boundary of an umbilic patch, connect them to the center of the umbilic area in order that they do not cross each other.

end

Algorithm 12: Extract crest/ridge lines on a 3D mesh by the method [CAZ 05d].

All the different methods described above give a set of crest/ridge lines. But in real-world applications, 3D meshes may be distorted, incomplete, rounded at edges or with different sampling density. Some resulting crest/ridge lines may be then non-significant as they are too short or too irregular. It is then required to add a post-processing step in order to filter these crest/ridge lines and keep only the significant ones. Several algorithms have been proposed, which are based on thresholding a parameter computed on the list of the points \mathbf{P}_i of a crest/ridge line:

In [OHT 04], the authors propose to use the *strength*:

$$Strength = \sum_j \|\mathbf{P}_{j+1} - \mathbf{P}_j\| \frac{k_{max}(\mathbf{P}_j) + k_{max}(\mathbf{P}_{j+1})}{2}$$

This parameter, which gives the average magnitude of the maximum principal curvature, is scale-independent. We can also find in [KIM 06] a formula which gives a slightly different parameter:

$$T = \sum_j \|\mathbf{P}_{j+1} - \mathbf{P}_j\|^2 k_{max}(\mathbf{P}_j)$$

In [CAZ 05d], the authors compute the Taylor expansion around a point \mathbf{P} of k_1 in the direction of its associated principal direction \mathbf{t}_1 . By starting with an extension of the equation [1.35] at order 4, which introduces in particular the term c_0x^4 , they obtain:

$$k_1(x) = k_1(\mathbf{P}) + b_0x + \frac{P_1(\mathbf{P})}{2(k_1(\mathbf{P}) - k_2(\mathbf{P}))}x^2 + O(x^3)$$

with $P_1(\mathbf{P}) = 6b_1^2 + (k_1(\mathbf{P}) - k_2(\mathbf{P}))(c_0 - 3k_1^3(\mathbf{P}))$.

For a crest/ridge point, we have by definition $k'(0) = 0$. But in plane or spherical areas, the principal curvatures are constant and this equality is also true. So, the idea is to study the second derivative $k''(\mathbf{P})$. The higher its value is (in absolute value), the faster the variation of curvature is. This means that for a maximum or a minimum of $k(\mathbf{P})$, the crest/ridge line will be sharp if $\|k''(\mathbf{P})\|$ is high.

By differentiating the previous equation, we get:

$$k_1''(x) = \frac{P_1(\mathbf{P})}{k_1(\mathbf{P}) - k_2(\mathbf{P})}$$

And we can define the *sharpness* of a crest/ridge line as:

$$Sharpness = L^2 \sum_j \left\| \frac{P_1(\mathbf{P}_j)}{k_1(\mathbf{P}_j) - k_2(\mathbf{P}_j)} \right\|$$

By multiplying by L^2 where L is the length of the crest/ridge line (or it could be more generally the mesh size as its diameter), we obtain a scale independent parameter to threshold.

In [YOS 08], the authors compute what they call the *cyclidity* by using two different formulas which gives similar results:

$$Cyclidity_1 = \sum_j \|\mathbf{P}_j - \mathbf{P}_{j+1}\| \sqrt{|e_{max}(\mathbf{P}_i)| + |e_{min}(\mathbf{P}_i)|}$$

$$Cyclidity_2 = L \sum_j \|\mathbf{P}_j - \mathbf{P}_{j+1}\| \sqrt{e_{max}^2(\mathbf{P}_i) + e_{min}^2(\mathbf{P}_i)}$$

where L is the length of the crest/ridge line and $e_{max}(\mathbf{P})$ (resp. $e_{min}(\mathbf{P})$) is the crest/ridge function for the blue (resp. red) crest/ridge lines (see section 1.4.3).

This parameter is also scale-dependent but as it involves third-order derivatives, it may be more complex to apply.

Some other methods to filter crest/ridge lines are based on a multi-scale analysis. In [SUB 99a], the idea is to use two versions of the 3D mesh, the original one \mathcal{M} and a smoothed version \mathcal{M}_s (many smoothing methods exist, we can refer to [TAU 00] for some supplementary information). On the smoothed version \mathcal{M}_s , we should extract longer and more reliable crest/ridge lines but the positions of their points may be shifted by the displacement of the 3D mesh vertices due to smoothing. Moreover, we can miss some crest/ridge lines emphasizing some small salient parts of the 3D mesh, which may have disappeared after the smoothing step. By registering the two sets of crest/ridge lines of \mathcal{M} and \mathcal{M}_s and keeping the common ones with the

position of points on M , we expect to keep the most reliable crest/ridge lines and while keeping their original accuracy. Nevertheless, the presented algorithm is quite crude because it uses only two values of the scale parameter that are manually defined. A more sophisticated multi-scale method can be found in [FID 97] where crest/ridge lines are extracted and followed across many scales.

Another method to get more significant results is to link several close short crest/ridge lines in order to get a longer one. In particular, crest/ridge lines are often cut around umbilic areas whereas they should cross them according to theory (see [CAZ 05d]). The first algorithm proposed in [PAG 06] is based on path planning widely used in robotics. The idea to link two crest/ridge short lines L_1 and L_2 is to define two artificial potential fields. The first field $U_{att}(\mathbf{P})$ will be based on the distance to L_2 , characterizing the proximity to a vertex of L_2 . The second field $U_{rep}(\mathbf{P})$ aims to repel or push according to the differential elements of the considered vertex. In our case, we could use, for example, an increasing function of $|e(\mathbf{P})|$. Then, the more this value is, the more the vertex \mathbf{P} can be considered as different of a crest/ridge point where e should be null. Now, if we start from an extremity \mathbf{P} of L_1 , we compute $U_{att}(\mathbf{P})$ and $U_{rep}(\mathbf{P})$ and we can define the vector \mathbf{g} corresponding to a weighted sum of the gradient vectors of the two fields. $\mathbf{P} + \alpha\mathbf{g}$ will define then a new point in the direction of the minimum path between L_1 and L_2 with respect to the potential fields. By iterating the process, we can link the two short crest/ridge lines. In [KHA 98], we can find the same idea of finding the optimal line linking two crest/ridge points by minimizing a criterion about the curvature of the 3D mesh. The method is based on dynamic programming but we could also use the very classic Dijkstra's algorithm to find the optimal path.

When detecting crest/ridge lines, the estimation of the differential parameters may be imprecise and may distort the positions of the points of the lines. It is then interesting to smooth these noisy crest/ridge lines. In [JIN 06], the authors propose to use an equivalent of the well-known Laplacian smoothing of 3D mesh: each point of the crest/ridge line will slide on the 3D mesh towards the middle of its previous and next point on the line. Compared to a direct Laplacian smoothing of the whole 3D mesh, this method concentrates more on the local shape of the crest/ridge lines and is more effective. Note that in [HIL 05], it is proposed to use the Hausdorff distance to guarantee that the smoothed line does not deviate strongly from the initial line but no detail is given.

In [JIN 06], the authors propose also to remove some crest/ridge branches which could be connected to long crest/ridge lines. For this purpose, they compute a Minimum Spanning Tree which allows them to identify the longest path. It then becomes possible to define automatically a length threshold and to disconnect small parts from the lines.

Finally, note that some methods to compute crest/ridge lines have been proposed when we have no more a 3D mesh but an unorganized 3D cloud of points \mathbf{P}_i .

In this case, the authors of [GUM 01] propose to connect all the segments between a 3D point of the cloud and its k -nearest neighbors in order to build a neighbor graph (a more complex method based on Delaunay tetrahedrization is also proposed). Then, for each 3D point, the 3×3 covariance matrix of the coordinates of the neighbor points is computed. The eigenvalues of this matrix give information about the local shape around the 3D point. In particular, it is possible to estimate a local fitting plane (and then a normal vector) and the curvatures in the direction of the neighbor points. Based on the eigenvalues of the endpoints, we can affect a weight to the edges of the neighbor graph. Then by extracting a Minimum Spanning Tree (and eventually removing short branches), we can define some feature lines. If we choose a weight characterizing if a point is locally a crease, we can extract crest/ridge lines.

In [MÉR 11], the covariance matrix is based on a 3D Voronoï diagram. The 3×3 matrix called the Voronoï Covariance Measure is approximated either by a Monte-Carlo algorithm or by a 3D discrete tessellation. All the VCM in the neighborhood of a point can be summed in order to get the convolved Voronoï covariance measure which can provide information about the normal vectors and the differential parameters of the surface which underlies locally the 3D point cloud. It is then possible to extract the so-called sharp edges that correspond to segments of crest/ridge lines. Note that the extraction of the covariance matrix can be multi-scale as proposed in [PAU 03] or [PAR 12].

In [ALT 13], the covariance matrix of the coordinates of the neighbor points of the point \mathbf{P}_i is used to build a local coordinate system where axes Ox and Oy correspond to the principal directions and the axis Oz to the normal vector. By interpolation, it is then possible to compute the height z of the local surface for the points x_l which are regularly sampled along Ox .

In order to get a robust and continuous representation of the curve $z(x_l)$, we can apply a discrete Fourier transform and remove the coefficients of higher frequency. By taking the curvature of the curve $z(x)$ at 0 (which can be directly derived from the Fourier transform), we get a robust value of one principal curvature (corresponding to the principal direction defining the axis Ox) at point P_i . By doing the same for the axis Oy , we can get all the differential parameters. Now, by thresholding the Gaussian curvature of the points P_i , we can select potential crest/ridge points. In a second step, an equivalent of Laplacian smoothing is applied to these potential points and when two of them are too close, they are merged. The smoothing process is iterated until the number of potential points remains constant; we can then consider to obtain a set of robust crest/ridge points. In the third step, crest/ridge lines are constructed by a line growing technique. To add a crest/ridge point to a current line, we take the two endpoints of the current line, we build two spheres centered at these endpoints and which radius is proportional to mean distance between a point P_i and its closest point and we search for the crest/ridge points inside the spheres. If one of these crest/ridge points is roughly aligned with the extremity part of the current line, it is added to the current line and the process can continue. The method has been implemented on GPU which allows us to process large point clouds.

1.4.4. Feature lines based on homotopic thinning

In this section, we present a way to characterize feature lines on a mesh using a well-known notion coming from mathematical morphology: the *homotopic thinning* [STE 71, ROS 75, ROS 82]. The feature lines are materialized by a skeleton computed by thinning a vertex set lying on the 3D meshes. The key idea comes down from eroding a 2D set located on a discrete 2-manifold. The main difficulty is to transpose the notion of neighborhood from the classical thinning algorithms where the adjacency is constant (e.g. 26-adjacency in digital volumes, 8-adjacency in 2D images) to the mesh domain where the neighborhood is variable due to the adjacency of each vertex. Kudelski *et al.* propose in their work a thinning operator dedicated to irregular 3D meshes in order to extract the skeleton of a vertex set, and thus feature lines corresponding to a specific criterion [KUD 11, KUD 13].

The skeleton is a popular and established shape descriptor. It is an entity that is globally centered in a 2D or a 3D object, and it characterizes its

topology and its geometry. This structure is widely used in various applications (video tracking [GAL 09], shape recognition [YU 08], surface sketching [MAR 09], etc.). Several techniques exist in order to extract the skeleton from binary 2D images [ZHA 84], 3D closed volume meshes [AU 08] or 3D cubic grids [LEE 94].

Nevertheless, very few approaches have been dedicated to the extraction of skeletons from a binary information located on an arbitrary 3D mesh. It remains to compute the skeleton of a skew subset of a discrete surface embedded in \mathbb{R}^3 . Rössl *et al.* [RÖS 00b] have presented a first method that uses the elementary *opening* mathematical morphology operator, ported to 3D triangulated meshes. However, the operator definition is not complete and the underlying algorithm presents some issues. Therefore, several drawbacks have been pointed out which mainly lead to unexpectedly disconnected skeletons [KUD 11]. Kudelski *et al.* have later proposed a modified algorithm that produces topologically robust skeletons, by generalizing the notion of *morphological erosion* to arbitrary 3D meshes [KUD 13]. This approach takes as an input a subset lying on a triangulated surface meshes in 3D, and outputs thin lines corresponding to the skeleton obtained by homotopic thinning. The main idea is to transpose the notion of neighborhood from the classical thinning algorithms where the adjacency is constant (e.g. 26-adjacency in digital volumes, 8-adjacency in 2D grids) to the mesh domain where the neighborhood is variable due to the adjacency of each vertex. The authors propose a thinning operator dedicated to irregular meshes in order to extract the skeleton of a vertex set.

Kudelski *et al.* work is interesting in the frame of feature line extraction because it carries the idea of homotopic thinning using a generalized adjacency. Non relevant vertices of the subset (topologically speaking, i.e. the *simple vertices*) are removed iteratively. It produces a lineal skeleton composed of initial vertices and edges. This skeletonization is general because it can deal with non-developable surfaces (operations are local, and there is no need to have a $[i, j]$ indexing like in 2D grids). Moreover, the resulting skeleton preserves the topology of the original shape lying on the surface: cycles and Y-junctions are completely preserved, and this is not the case of previously described approaches where umbilic points create discontinuities of the feature lines.

Let \mathcal{M} be an unstructured mesh patch representing an arbitrary manifold surface \mathcal{S} , such as $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$. The sets \mathcal{V} , \mathcal{E} and \mathcal{T} correspond, respectively, to the vertices, the edges, and the triangles composing \mathcal{M} , a piece-wise linear approximation of \mathcal{S} . We denote p_i the vertices, with $i \in [0; n[$ and $n = |\mathcal{V}|$ being the total number of vertices of \mathcal{M} . The neighborhood \mathcal{N} of a vertex p_i can be defined as follows:

$$\mathcal{N}(p_i) = \{q_j \mid \exists \text{ a pair } (p_i, q_j) \text{ or } (q_j, p_i) \in \mathcal{E}\}. \quad [1.39]$$

In such a case, $m_i = |\mathcal{N}(p_i)|$ represents the total number of neighbors of p_i .

Definition of the subset R

As we consider obtaining a skeleton of a subset of \mathcal{M} , let us now define a binary attribute F on each vertex of \mathcal{V} . The set $R \subseteq \mathcal{V}$ is then written as follows:

$$\forall p_i \in R \iff F(p_i) = 1. \quad [1.40]$$

The attribute F may be defined from a previous process such as a manual selection, a thresholding based on geometric properties (triangle area, principal curvatures, etc.) or any binarization process. Then, an edge $e = (p, q)$ belongs to R if and only if $p, q \in R$. Similarly, a triangle $t = (p, q, r)$ belongs to R if and only if $p, q, r \in R$.

In [KUD 13], the main objective was to develop a technique to extract the skeleton of the set R by using a topological thinning based on the mesh connectivity. This skeletonization algorithm consists of an iterative thinning, relying on a classification of each vertex of R . The authors proposed four vertex types based on $c(p_i)$, the *complexity* of the vertex p_i defined such as:

$$c(p_i) = \sum_{j=0}^{m_i-1} |F(q_j) - F(q_k)|, \quad [1.41]$$

where $k = j + 1 \bmod m_i$ and $q_j, q_k \in \mathcal{N}(p_i)$.

A vertex p_i is said to be *complex* if and only if $c(p_i) \geq 4$. The set of all *complex* vertices is named C . A *complex* vertex p_i thus potentially corresponds

to a part of a skeleton branch if $c(p_i) = 4$, or a connection through several branches if $c(p_i) > 4$.

A vertex p_i is said to be *center* if and only if $\mathcal{N}(p_i) \subseteq R$. The set of all *center* vertices is named E .

A vertex p_i is called *disk* if and only if $\exists q_j \in \mathcal{N}(p_i), q_j \in E$ that is a *center*. The set of all *disk* vertices is named D . A *disk* vertex corresponds to a *simple* vertex: a point that does not modify the expected skeleton's topology if it is removed (by analogy with *simple points* [BER 96]).

A vertex p_i is marked as *outer* if and only if $F(p_i) = 1$ and $p_i \notin (C \cup D \cup E)$. The set of *outer* vertices is named O and is defined as follows:

$$O = R \setminus (C \cup D \cup E) \quad [1.42]$$

The algorithm then removes one by one all the *disk* vertices that are not converted to a different class after the thinning operator application. Indeed, at each iterative thinning step, a *disk* vertex may change from one class to another and, as a side-effect, this may lead to potential disconnections during the skeletonization. To counteract this issue, this requires specifying explicitly a verification stage in the algorithm: at each application of the thinning operator, the class of a vertex is recomputed before its deletion. For example, if a *disk* vertex becomes a *complex* vertex, the vertex is not removed.

After applying the skeleton operator until idempotency on R , the set of the remaining vertices, corresponding to the final *skeleton*, is called Sk_R . During each pass, the skeleton operator removes the boundary *disk* vertices (if they do not change class). The thinning approach can be summarized by the following algorithm 13.

Figure 1.8 illustrates the execution of the algorithm. The extracted skeleton is fully connected and faithfully characterizes the topology of R

After obtaining the skeleton Sk_R of R , it is possible to remove the smallest branches by running a last and optional *pruning* operation (like in most of the skeleton approaches).

Input: Region R defined from a subset of vertices on a mesh

Output: Skeleton Sk_R of the region R

```

begin
  Repeat until idempotency:
    forall vertex  $p_i \in R$  do
      if  $p_i$  is a disk vertex then
        Compute the complexity of vertex  $p_i$ 
        if the class of  $p_i$  does not change then
          | Remove  $p_i$  from  $R$ 
        end
      end
    end
  end
end

```

Algorithm 13: Thinning algorithm.

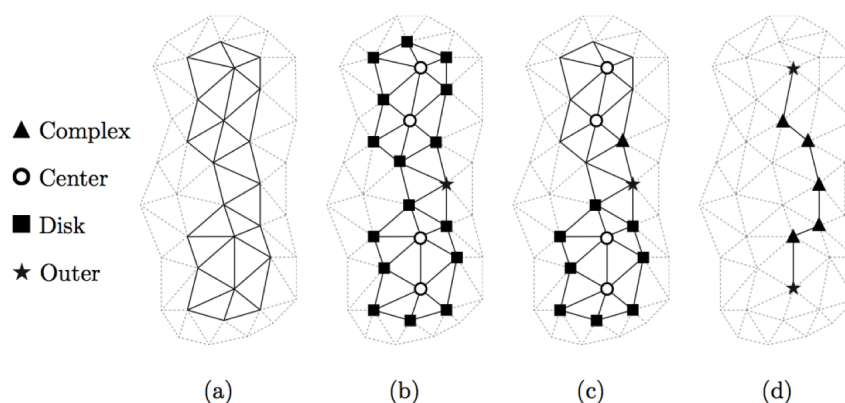


Figure 1.8. Illustration of the homotopic thinning algorithm based on a generalized adjacency (Kudelski et al. [KUD 11, KUD 13]): (a) region R , (b) vertex classification, (c) execution of the thinning algorithm with update and (d) final skeleton fully connected

The main asset of this approach is that the obtained skeletons describe the geometry and the topology of the original set R . For instance, in Figure 1.9, the algorithm has been tested on irregular meshes to show the robustness of the proposed approach. It can be noted that the resulting skeletons are the expected ones and reflect correctly the topology and geometry of the original set R in a proper way.

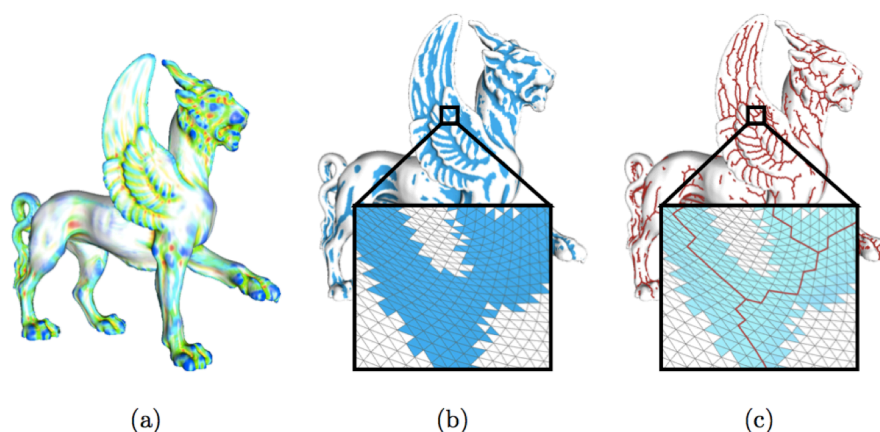


Figure 1.9. Algorithm of feature line extraction: (a) curvature estimation, (b) definition of the set R and (c) extraction of lines from R by Kudelski et al. thinning approach (courtesy of [KUD 11, KUD 13]). For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

Kudelski *et al.* approach produces topologically robust skeletons, by generalizing the notion of *morphological erosion* to arbitrary meshes. This method outperforms approaches based on the study of the variation of curvatures (see, for example, [YOS 08]), because there is no need to compute third-order estimators (the noise is thus handled in a better way) and the umbilic points can be processed without cutting the lines. No post-processing is needed to obtain X-junctions on surface meshes, which makes this approach robust, efficient and contrasting.

1.5. Region-based approaches

1.5.1. Mesh segmentation

Segmenting areas on a surface model consists of defining a partitioning of the surface into several regions, according to a specific criterion. Such a criterion can be related to the area of a region, curvatures, flatness, but also to semantic aspects of the shape (arm, leg, face of a human body, for example).

Segmenting and detecting feature lines are generally dual problems: once the regions of a segmentation are determined, their boundaries can be

considered as feature lines. Conversely, feature lines can sometimes be assembled so that they bound the regions of a segmentation.

The techniques dealing with mesh segmentation are numerous and varied. Many comparative studies have been published on this topic (see, for example, [ATT 06b, AGA 07, SHA 08a, THE 15, ROD 18]).

Among the different kinds of approaches, region growing is the easier way to handle segmentation on meshes. It consists of an iterative process starting from a given number of points called *seeds* defined on the surface. Vertices of the 3D mesh are then added to a seed region if they are adjacent to this region and if they match a particular criterion, such as curvature [LAV 05] (see Figure 1.10) or flatness [KAL 96].

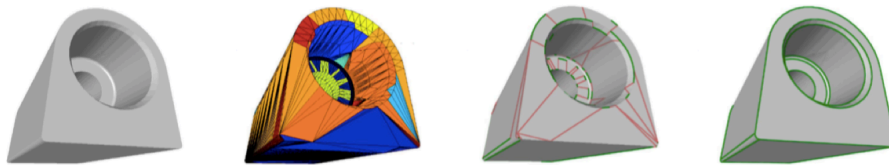


Figure 1.10. *Segmenting approach proposed by Lavoué et al.. The segmentation is processed by a region growing technique constrained by a curvature criterion. The boundaries of the regions are first extracted from the segmented areas, then corrected and completed. Image from [LAV 05]. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip*

Mesh segmentation by partitioning is similar to region growing. Approaches in this category aim at expressing the properties of the elements of the mesh within a common space of representation, in order to define subsets by using a classification algorithm. As always, the choice of the discriminating property is crucial. For instance, it is possible to use membership probabilities [KAT 03] (see Figure 1.11), an approximation with primitives [ATT 06a] or a distance function [SHL 02].

In the domain of image processing, *watersheds* are commonly used in segmentation. This notion has been extended to 3D mesh segmentation. However, the main difficulty remains in defining a relevant height function on a geometric structure, which is not an elevation model. In order to artificially add a scalar value on each vertex of the mesh and thus to use a watershed-like

approach based on heights, the curvature [DEL 06] or the dihedral angle [ZUC 02] can be used.



Figure 1.11. Segmentation based on membership probabilities. Image from [KAT 03]. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

In addition to the aforementioned methods, one family of approaches derives from spectral analysis. They shift the mesh segmentation problem to a matter of planar graph partitioning. The mesh segmentation comes from the eigenvalues of an affinity matrix [LIU 04] (see Figure 1.12) or a matrix of geodesic distances between the vertices of the graph [ZHO 04].

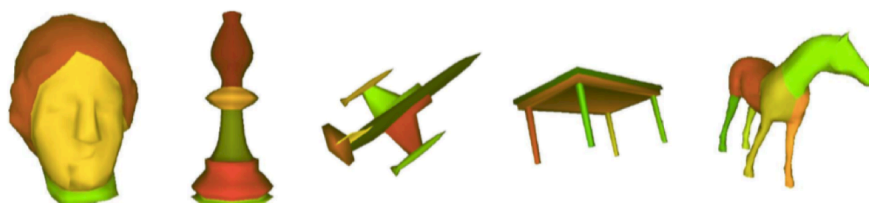


Figure 1.12. Segmentation using spectral analysis. Image from [LIU 04]. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

A curve called *skeleton* can sometimes be used to guide the segmentation process: Shapira *et al.* define a *shape diameter function* that provides a relevant description of the volume of the model, in order to characterize the boundaries between the different parts of the object [SHA 08b].

Some approaches are based on statistics: their aim is to classify data, in order to emphasize the relations that could exist between them. [BEN 11] present a segmentation method based on a heat flow mapping. The starting points are initialized according to the curvature and to the connectivity of the

vertices of the mesh. [GOL 08] developed a randomized segmentation approach: starting from the original mesh, several segmentations are performed to produce one final segmented object.

Becoming more and more popular are machine learning techniques, which can also be used to enhance the segmentation process, as soon as a large database of pre-segmented meshes is already available. For instance, [KAL 10] propose a data-driven approach to simultaneous segmentation and labeling of parts in 3D meshes. An objective function is learned from a collection of pre-labeled training meshes. This function is formulated as a Conditional Random Field model, with terms assessing the consistency of faces with labels.

1.5.2. Shape description based on graphs

Graphs have been used in order to describe shapes and layout of sub-shapes mostly within an image, i.e. in 2D. For instance, Damiand *et al.* use combinatorial maps to describe image partitions within a generic segmentation algorithm [DAM 12].

Some recent studies are based on the division of 3D shapes into overlapping regions determined by their regularity properties, such as symmetries [TEV 14]. Then a graph is formed that connects the pieces with pairwise relations that capture geometric relations between rotation axes and reflection planes. Finally, the authors perform graph matching to establish correspondences.

Graph and sub-graph processing for the analysis of shapes is a known methodology used for instance to detect shape symmetries [BER 08], to extract features on points clouds using sub-graphs selection [GUM 01], to perform mesh segmentation [ZHA 08a]. Approaches using “skeleton graphs” have been developed to achieve global matching of shapes [SUN 03] or to locally match the surface parts using discrete curvature as an invariant descriptor [GAL 06]. We now describe in detail one of them to give an example.

Polette *et al.* have recently developed an approach aiming at segmenting a mesh according to several areas with the same “differential type” (using the same classification table with eight categories) [POL 15b, POL 15a, POL 17].

It consists of decomposing the surface into patches, and in constructing an underlying graph describing the shape. Each node is associated with an area of the same type.

The table (see Figure 1.13) described by Besl *et al.* [BES 88b] categorizes a vertex among eight different types according to the signs of the mean curvature H and the Gaussian curvature K : *peak*, *ridge*, *saddle ridge*, *minimal*, *saddle valley*, *valley*, *pit* and *flat*. This table is used to tag the areas of a surface.

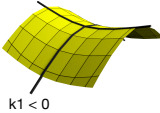
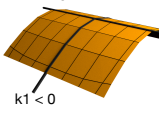
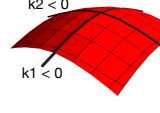
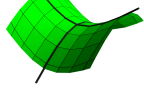
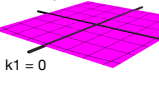
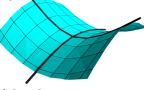
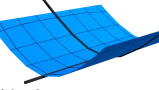
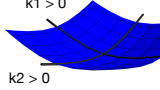
	$K < 0$	$K = 0$	$K > 0$
$H < 0$	<p>saddle ridge</p> <p>$k_2 > 0$</p> <p>$k_1 < 0$</p> 	<p>ridge</p> <p>$k_2 = 0$</p> <p>$k_1 < 0$</p> 	<p>peak</p> <p>$k_2 < 0$</p> <p>$k_1 < 0$</p> 
$H = 0$	<p>minimal</p> <p>$k_1 = -k_2$</p> 	<p>flat</p> <p>$k_2 = 0$</p> <p>$k_1 = 0$</p> 	
$H > 0$	<p>saddle valley</p> <p>$k_1 > 0$</p> <p>$k_2 < 0$</p> 	<p>valley</p> <p>$k_1 > 0$</p> <p>$k_2 = 0$</p> 	<p>pit</p> <p>$k_1 > 0$</p> <p>$k_2 > 0$</p> 

Figure 1.13. Shape categories using mean and Gaussian curvatures.
For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

However, depending on how a surface is triangulated, a same shape can have several divisions. Thus, it can lead to several different graphs for one similar shape, due to the discrete aspect of the representation by meshes. To tackle this, the main idea is to proceed by analogy with the continuous world: when dealing with a continuous C^2 surface, it is not possible to go from a bumped area (with Gaussian curvature $K > 0$ and mean curvature $H < 0$) to a saddle vertex (with Gaussian curvature $K < 0$) without passing a $K = 0$ vertex (like a ridge or a valley). Meshes are discrete surface representations, and they can host two adjacent vertices with two non-adjacent types: one peak vertex can be the immediate neighbor of a

saddle or a pit vertex. Because this statement cannot happen for continuous surfaces, Polette *et al.* added *intermediary patches* to ensure consistency at the transition between areas.

Graph construction

Figure 1.14 presents an overview of the graph construction method. Starting from a triangulated mesh on which each vertex has been labeled according to its class (among the eight possible categories based on Gaussian and mean curvatures, respectively K and H), the shape is then decomposed into patches of the same type. Subsequently, transition areas are defined at a sub-vertex level, and next the graph is constructed.

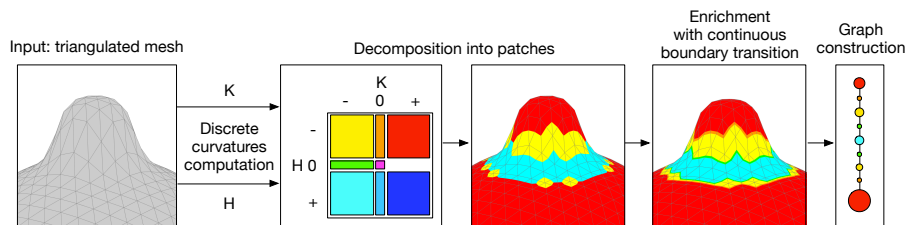


Figure 1.14. Overview of the graph construction methodology. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

Polette *et al.* use Meyer *et al.*'s discrete curvature estimator [MEY 03] to obtain the mean curvature H and the Gaussian curvature K . It is a robust curvature estimation based on Voronoï cells and a finite-element method.

After this labeling, a mean filter is applied to the discrete surface, in order to smooth and remove irrelevant small areas that do not correspond to feature zones on the shape. A mathematical morphology filter with several distances is employed on the mean and Gaussian curvatures values (see Figure 1.15).

In order to regularize the patches computed on the mesh, because discrete curvatures estimators depend on the sampling, intermediate patches are added to the layout. If the mesh was derived from a continuous object, then it would be impossible to have a peak patch connected to a pit patch without having a transitional patch. Thus, continuity rules are defined to ensure a consistency between all the different patches. Figure 1.16 shows, for each patch status,

the compatible patches that can be adjacent. If a yellow area (saddle ridge) is connected to a red area (peak), then there must be an orange area (ridge) in between.

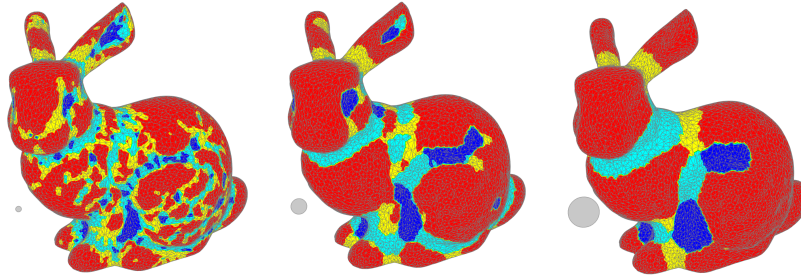


Figure 1.15. Influence of the distance parameter on the bunny mesh (courtesy of the Stanford University Computer Graphics Laboratory); the radius distance is shown as a gray disk. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

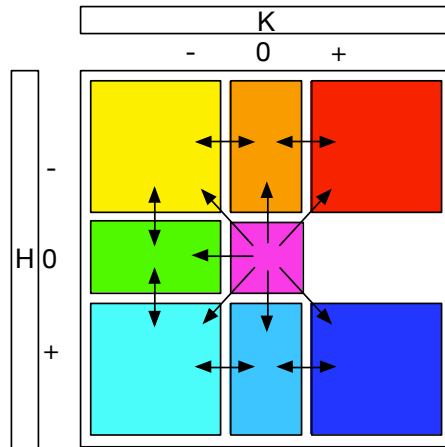


Figure 1.16. Adjacency rules between patches. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

To induce these transition areas, Polette *et al.* use an implicit interpolation of the curvatures between patches to establish the tolerated adjacency for each category. Using these rules, only one shortest path exists between two

patches. Figure 1.17 shows a mesh with the addition of continuous transition boundaries. Each patch between two red patches forms a ring. The yellow parts are thus linked and considered to be a unique area.

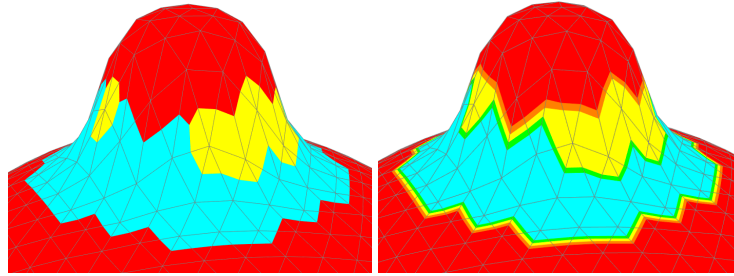


Figure 1.17. Continuous boundary incorporation on a simple bumped mesh. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

The sub-vertex adding approach is illustrated in Figure 1.18. Each triangle of the mesh is checked to ensure that the adjacency rules are respected for all edges. If not, then new areas are added between them and linked to their neighbors. The first two columns describe the adding of missing nodes. Before adding a new node, its existence is checked. If the node does not exist, then a new one is built; otherwise, the existing one is used. Then links between the additional nodes are created (see columns 2 and 3).

After the sub-vertex areas adding step, a graph is constructed using the patch neighborhood. A propagation algorithm is used to select all contiguous vertices and build a list of patches by category. A node of the graph is defined for each patch. Each node holds the category, the patch area and a link to each neighboring patch (see Figure 1.19).

In Figure 1.20, three meshes from the same shape with different samplings are presented, in order to show how the boundary adding affects the graph's consistency. The sub-vertex step helps producing three similar graphs, even though the computed curvature areas have different layout and sizes.

Feature extraction

Polette *et al.* apply the shape description graph defined above to feature extraction. It can be integrated into several applications, like semantic feature

description, similarities extraction between meshes or self-similarities retrieval within one single mesh.

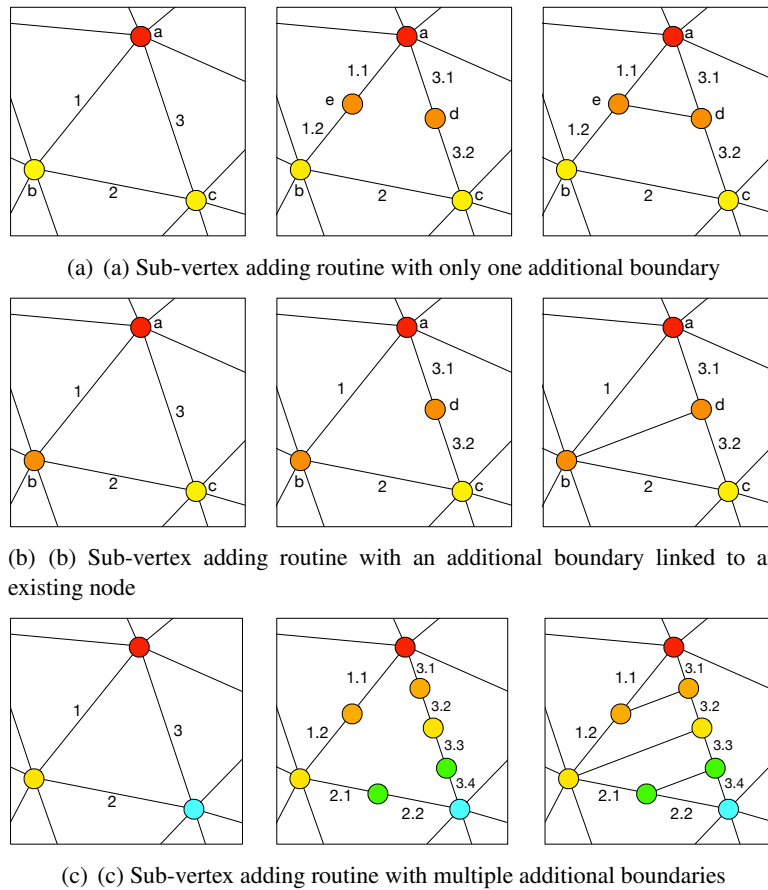


Figure 1.18. Continuous boundary adding. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

As a graph can describe a 3D shape, a specific feature within the shape can be described by a sub-graph. Three strategies are proposed by Polette *et al.* to extract features according to different purposes: using a hand-made pattern, using two input meshes or using one single mesh as input to extract self-similarities [POL 15a, POL 17].

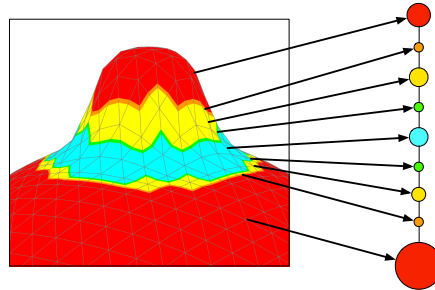


Figure 1.19. Graph construction procedure. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

Extracting patterns from a shape descriptor graph consists of a partial sub-graph matching problem. Because the graphs at stake have different node categories and sizes, with specific adjacent rules, Polette *et al.* have developed a specific matching approach in three steps: construction of a similarity matrix S , selection of the starting node pairs and recursive node pairing from starting pairs.

The similarity matrix is constructed based on the method described by Nikoli [NIK 12]. This approach is designed to compute the similarity between two entire graphs and Polette *et al.* have adapted it so it can locally compute the similarity between two nodes.

Let us consider two graphs G_1 and G_2 as inputs defined as $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where V is a set of nodes and E is a set of edges. $|V_1|$ and $|V_2|$ represent the number of nodes in each graph. The *similarity matrix* S is a $|V_1| \cdot |V_2|$ matrix, with S_{ij} the similarity between the nodes V_{1i} and V_{2i} , $S_{ij} \in \mathbb{R}$ and $0 \leq S_{ij} \leq 1$. Two identical nodes give a similarity value of 1, and two strictly different nodes give a value of 0. The criterion to construct the matrix is the following: “two nodes $i \in VA$ and $j \in VB$ are considered to be similar if neighbor nodes of i can be matched to similar neighbor nodes of j ” (see [NIK 12]).

To extract shared sub-graphs, recursive node pairing is operated. Starting pairs are found by testing the similarity value using a threshold t . A ij pair is a starting pair if $S_{ij} > t$. Starting from each selected pair, each node of their neighborhoods is recursively paired by the maximum similarity value. This pairing function is recursively called on each new pair ij if $S_{ij} > t$. The maximum size of each ij pair of paired sub-graphs is saved in a new $|V_1| * |V_2|$ matrix M .

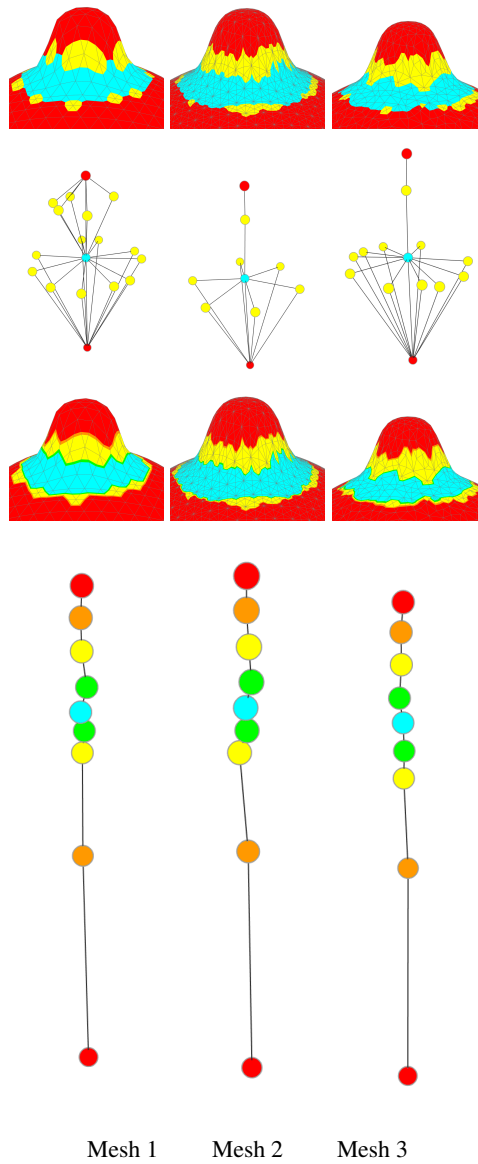


Figure 1.20. Three examples of graph construction with (bottom) and without (top) continuous boundary adding on a similar shape with different sampling. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

Maximum similar patterns are extracted using the matrix M , beginning with the maximum values of M_{ij} . Similar sub-graphs are detected using the same previous recursive method. Corresponding pairs are indexed as a new extracted pattern. Multiple similarities can be detected by propagating this index to all the same maximum values through the rows and columns of each indexed node.

Some examples of application

Examples are presented in this section to illustrate the use of Polette *et al.* shape descriptor graphs for each scheme.

Semantic description of a feature

A specific feature can be described semantically via a pattern. Figure 1.21 shows the characterization of a feature on the wing of the *gargoyle*¹² mesh. The sub-graph used in Figure 1.21(d) can describe semantically “a pit bounded by a saddle ridge that can contain one or more peaks, the whole area being bounded by a saddle valley”.

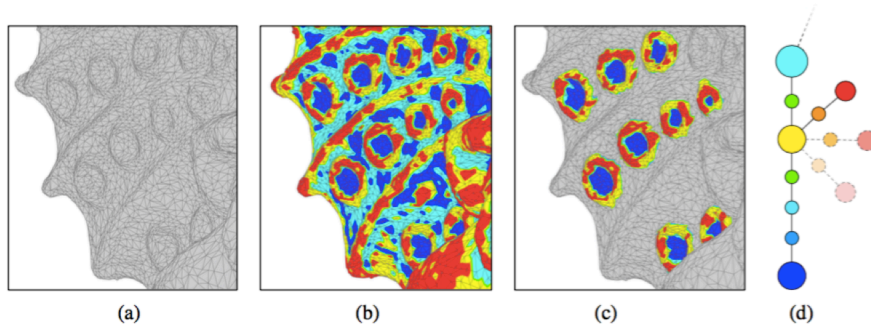


Figure 1.21. Feature extraction by terminal sub-graph recognition: (a) input mesh, (b) mesh partition into patches, (c) extracted features and (d) terminal sub-graph used as a feature descriptor. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

¹² Courtesy of the AIM@SHAPE consortium.

In Figure 1.22, Polette *et al.* use this approach to detect lotus flowers in the Buddha mesh¹³ (this detection can also be found in [GAL 06]). The yellow part of the flowers and the branches belong naturally to the same node. By limiting the size of patches, the authors can extract only the flowers.

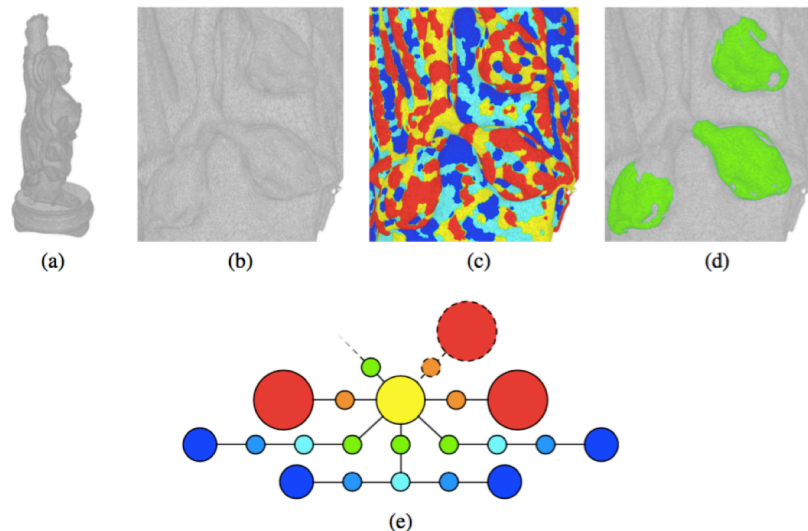


Figure 1.22. Extraction of flowers on the Buddha mesh (b), categories (c), extracted parts (d) and the extraction pattern used (e). For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

Similarity between two meshes

The authors illustrate this approach can be used to detect similarities between two or more meshes. The Figure 1.23 presents two extracted features, a peak and a pit, that can be found on two different meshes.

Self-similarity within a mesh

Self-similarity within a single mesh can be performed with the shape descriptor graphs by adding an additional constraint in order to avoid the trivial pairing of all nodes to themselves: a node cannot be paired to itself in the final pairing procedure.

¹³ Courtesy of the Stanford University Computer Graphics Laboratory.

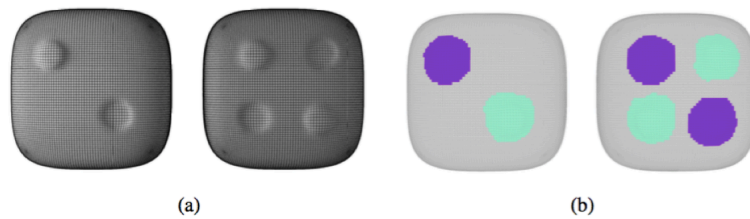


Figure 1.23. Feature detection between two meshes: (a) input meshes and (b) extracted features. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

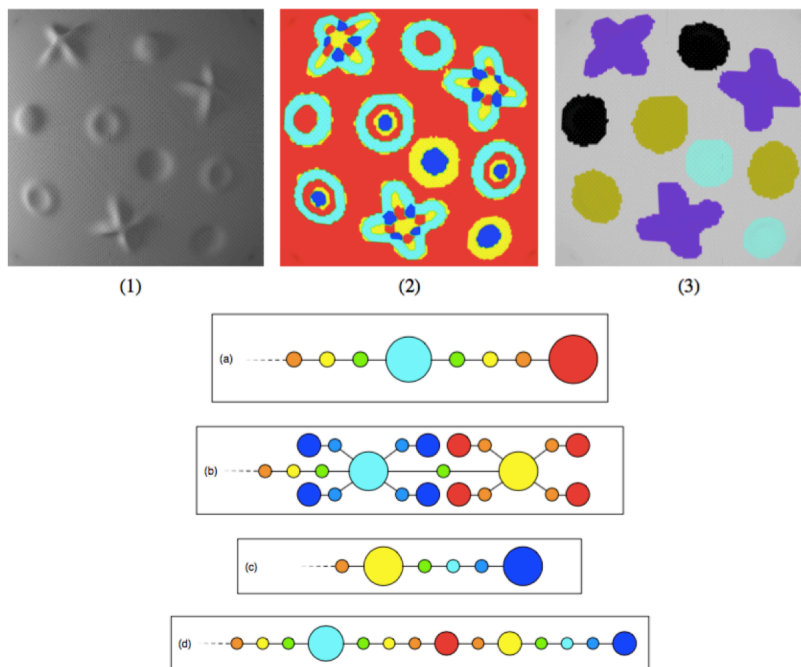


Figure 1.24. Self-similarity extraction on a mesh: (1), (2) and (3) show the input mesh, the computed list of patches and the output classification of sub-graphs. Extracted sub-graphs produced by the approach are presented in (a), (b), (c) and (d). For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

Figure 1.24 presents an example of self-similarity detection on a mesh. Four maximum sub-graphs are found multiple times; as an additional output, the method gives the sub-graphs that characterize a peak (a), a cross (b), a pit (c) and a crater (d).

Among mesh segmentation and feature zones detection approach, the method recently developed by Polette *et al.* is interesting for several reasons. It provides a simple graph formalism to describe 3D shapes, based on a curvature map and using a connecting graph. Features can be detected by extracting relevant sub-graphs that correspond to specific patterns. It can be integrated to several applications, as semantic feature description, similarities extraction or self-similarities detection.

1.6. Conclusion

In this chapter, we have presented some geometric features which characterize the shape of a 3D mesh. We focused specifically on features based on differential geometry parameters as they allow us to define not only particular points (umbilics) but also lines (parabolic lines, curvature lines or crest/ridge lines) as well as regions. Moreover, all these features are mathematically related: curvature lines or crest lines end on umbilics, regions are delimited by parabolic lines. This allows us to define a schematic modeling of the shape of the 3D mesh as it was proposed more than 35 years in [HAR 83]. We have seen that this modeling could be represented by a graph. Nevertheless, the discretization of the 3D mesh, and then the approximation of differential parameters, makes in general this graph quite unstable with respect to some small variation of the coordinates of the vertices. It is then interesting to define some global features of the 3D mesh which could stabilize this geometric modeling. In the next chapter, we present some features which are based on topology, which by definition, does not change when the shape is continuously deformed.

Topological Features

This chapter focuses on features defined from a topological point of view. *Topology* is a branch of mathematics concerned with the *qualitative* analysis of the main properties of spaces. In particular, topology is interested in features that do not change when the space is continuously deformed, without any tearing or gluing. Contrary to geometry, topology is interested in the global characteristics of spaces and does not care for measures and distances. The modern development of topology roots back to Henri Poincaré who introduced in 1895 the concept *homology*, which we will focus on later, although previous well-known mathematicians such as Gottfried Leibniz and Leonhard Euler had started studying objects and spaces globally in the 17th and 18th Centuries respectively.

The chapter is organized as follows:

- 1) in section 2.1, we browse useful mathematical concepts about surface topology and define what are the topological features of a surface;
- 2) in section 2.2, we review methods to compute such topological features;
- 3) in section 2.3, we explain how geometric information can be used to sort or filter topological features or vice versa.

2.1. Mathematical background

Topology enables us to properly define a continuous or discrete *surface*. In this section, we summarize the main concepts leading to such definitions and to a fundamental surface classification theorem (section 2.1.1). We then

study discrete surfaces from an algebraic point of view. This enables us to define more topological features for such surfaces, particularly meshed surfaces (section 2.1.2).

The concepts that we briefly mention here are detailed in several books. The reader interested in this topic can refer to [MUN 84, HAT 02, EDE 06, BIA 14], for example.

2.1.1. A topological view on surfaces

The primary concept in topology is the one of *topological space*. A *topological space* is an ordered pair (\mathbb{X}, X) , where \mathbb{X} is a set and X is a collection of subsets of \mathbb{X} such that:

- the empty set \emptyset and \mathbb{X} itself belong to X ;
- any finite or infinite union of elements of X belongs to X ;
- the intersection of any finite number of elements of X belongs to X .

X is called a *topology* on \mathbb{X} and its elements are called *open sets*. For example, for any $d > 0$, the Euclidean space \mathbb{R}^d is a topological space. Its open sets are the open balls $B_{x,\varepsilon} \forall x \in \mathbb{R}^d, \forall \varepsilon > 0$, i.e. the sets of points in \mathbb{R}^d whose Euclidean distance from x is smaller than ε .

We now drop the collection of subsets X and will only refer to \mathbb{X} as a topological space. The open sets of \mathbb{X} enable us to define the *neighborhood* of any point $p \in \mathbb{X}$: a *neighborhood* of p is simply an open set containing p .

From the notion of topological space, we reach the fundamental notion of *manifold*. A topological space \mathbb{X} is called a *k-manifold*, $k \geq 0$, if for any point $p \in \mathbb{X}$ there exists a neighborhood $V \subset \mathbb{X}$ that is homeomorphic to \mathbb{R}^k . In other words, there exists a function $f : V \xrightarrow{\sim} \mathbb{R}^k$ that is bijective, continuous and whose inverse is continuous. A topological space \mathbb{X} is called a *k-manifold with boundary*, $k \geq 0$, if for any point $p \in \mathbb{X}$ there exists a neighborhood $V \subset \mathbb{X}$ that is homeomorphic to \mathbb{R}^k or \mathbb{H}^k , where \mathbb{H}^k is the k -dimensional half-space of \mathbb{R}^k : $\mathbb{H}^k = \{x = (x_1, \dots, x_k) \in \mathbb{R}^k, x_1 \geq 0\}$. The *boundary* of \mathbb{X} is the subset of points of \mathbb{X} with a neighborhood homeomorphic to \mathbb{H}^k . It is either a $(k - 1)$ -manifold (without boundary) or the empty set.

In topology, the usual definition of a *surface* is the one of a 2-manifold. As a result, a topological surface is such that the neighborhood of any point is homeomorphic to a disk. Similarly, a *surface with boundary* is defined as a 2-manifold with boundary. The neighborhood of any point is homeomorphic to a disk or a half-disk. Figure 2.1 shows some examples of topological surfaces. Figure 2.2 shows some examples of **non-manifold** objects.

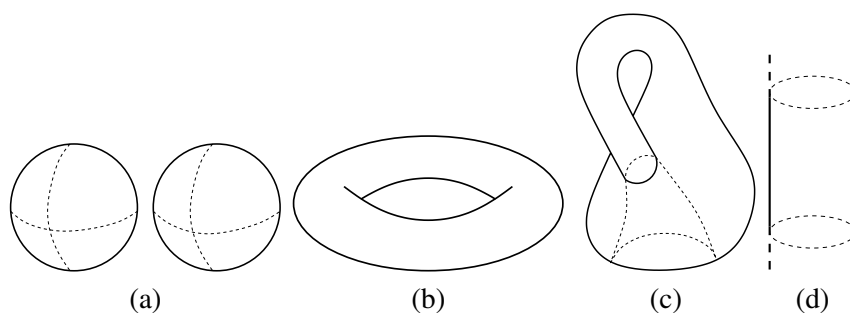


Figure 2.1. Some examples of topological surfaces. (a) A non-connected surface (two connected components). (b) A surface with non-zero genus: the torus. (c) A non-orientable surface: the Klein bottle. (d) An unbounded surface: the infinite cylinder

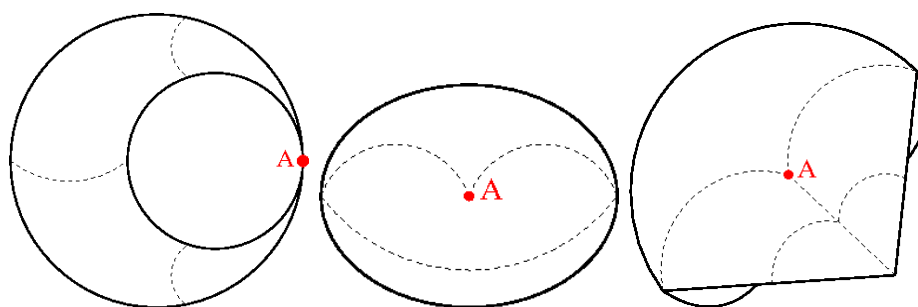


Figure 2.2. Some examples of non-manifold objects. In all cases, the neighborhood of the point *A* is neither a disk nor a half-disk. More examples can be found in [LÉO 09]

Note that so far we have not talked about metrics nor distances. In fact, the definition of a topological surface does not depend on any higher-dimensional space in which the surface “lives”. However, since now we need to restrict to *bounded* surfaces, we have to properly define such surfaces as subsets of

Euclidean spaces. An *embedding* of a k -manifold \mathbb{X} into \mathbb{R}^n , $n \geq k$, is a map $f : \mathbb{X} \hookrightarrow \mathbb{R}^n$ that yields a homeomorphism between \mathbb{X} and $f(\mathbb{X})$. If such a map exists, \mathbb{X} is said to be *embedded* into \mathbb{R}^n . In this book, we are particularly interested in 2-manifold surfaces embedded into \mathbb{R}^3 .

The great interest of topology is that it allows us to **classify** surfaces. Recall that a *compact subset* of the n -dimensional Euclidean space \mathbb{R}^n is a subset that is closed (meaning its complement in \mathbb{R}^n is an open set) and bounded (meaning it is contained in a Euclidean ball with finite radius). A subset of a surface is a *connected set* if it cannot be represented as the union of at least two disjoint non-empty open subsets. The maximum connected subsets of a surface are called the *connected components* of the surface. The number of connected components is the **first topological feature enabling us to classify surfaces**. For example, surfaces shown in Figure 2.1 (b,c,d) have one connected component, while the surface shown in Figure 2.1 (a) has two. The **second topological feature** of a surface is called its *genus* and derives from the fundamental *surface classification theorem*: any compact, connected 2-manifold (without boundary) is homeomorphic to either a sphere with g holes, $g \geq 0$ or a sphere with g cross-caps, $g \geq 1$ ¹. A surface homeomorphic to a sphere with g_1 holes cannot be homeomorphic to a sphere with g_2 holes if $g_1 \neq g_2$ nor to a sphere with g_2 cross-caps even though $g_1 = g_2$. g is called the *genus* of the surface. A surface homeomorphic to a sphere with g holes can be embedded in \mathbb{R}^3 and is called *orientable*, whatever the value of g . A surface homeomorphic to a sphere with g cross-caps cannot be embedded in \mathbb{R}^3 and is called *non-orientable*, whatever the value of g .

A *cross-cap* (see Figure 2.3) is a surface with boundary that is constructed by attaching a Möbius strip to the boundary of a disk. It has only one face and intersects itself along an interval.

Note that an equivalent theorem exists for 3-manifolds: any compact, simply connected 3-manifold is homeomorphic to the 3-sphere. This theorem, although much simpler in its formulation, was stated by Henri Poincaré in 1904 but proved only 100 years later by Grigori Perelman [PER 02, PER 03a, PER 03b]. It is known as the Poincaré conjecture.

¹ A 2-manifold with both holes and cross-caps is homeomorphic to a 2-manifold with only cross-caps, each hole being equivalent to two cross-caps in this case.

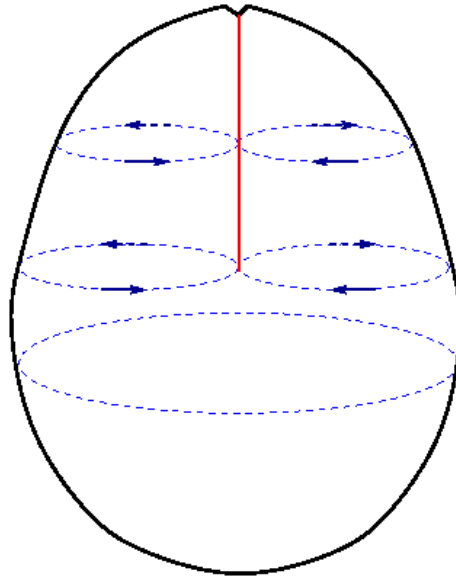


Figure 2.3. A view of the real projective plane, the compact non-orientable surface that is obtained by gluing a cross-cap to a disk. The self-intersection interval of the cross-cap is shown in red. The non-orientable surface obtained by gluing two cross-caps together at their boundary is the Klein bottle (see Figure 2.1 (c)). For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

2.1.2. Algebraic topology

In 1895, Henri Poincaré understood that additional tools would be necessary to better understand and characterize topological spaces. He proposed to let these spaces, especially manifolds, be added or multiplied by a scalar [POI 95]. This way, he introduced algebra into topology. For example, a manifold can be decomposed into a coherent collection of *simplices*, i.e. vertices, edges and triangles. Algebra is introduced on simplices by giving an *orientation* to each edge and each triangle, which is defined as a linear ordering of their vertices. Poincaré named *homology* this new concept of computations over topological spaces, in order to emphasize that he was interested in studying topological spaces sharing features, albeit not being strictly equal from a geometric point of view.

The features studied by homology are closed loops of dimension k for any k lower than the dimension of the topological space. These loops are called

k -cycles. Among such cycles, homology is more specifically interested in the equivalence classes of cycles which are *not* the boundary of a $k + 1$ -dimensional subset of the topological space called a *chain* (see section 2.3.1 for more details). Homology is described in terms of *homology groups*. Each *homology group* $H_k(\mathbb{X})$ of a topological space \mathbb{X} contains the equivalence classes of the k -cycles of \mathbb{X} which are not a boundary. Let us detail the example of the 2-sphere. 0-cycles are points. No point is the boundary of a line (1D subset of the sphere) on the sphere because a line has either two boundary points or none (in case it is a loop). All points on a sphere are homologous (i.e. belong to the same equivalence class for homology) roughly speaking because any of them can be moved to any other on the sphere along a curve belonging to the sphere. This is because the sphere is connected. Thus, $H_0(\mathbb{X})$ contains one equivalence class. $H_1(\mathbb{X})$ does not contain any equivalence class since any cycle made up of lines is the boundary of a 2D connected subset of the sphere. Since the sphere is of dimension 2, it has no subset of dimension 3, and thus no surfacic cycle is a boundary. All cycles being actually equivalent, $H_2(\mathbb{X})$ also contains one equivalence class. In the case of a torus, $H_0(\mathbb{X})$ and $H_2(\mathbb{X})$ also contain one equivalence class each, but this time $H_1(\mathbb{X})$ contains two equivalence classes, which are represented by two representative cycles in Figure 2.4 (b). Note that none of these two cycles is the boundary of a submanifold of the torus, since such submanifold would require additional lines as boundaries.

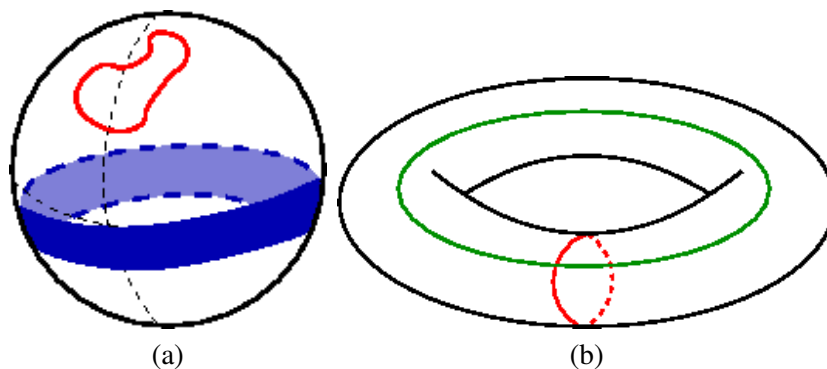


Figure 2.4. (a) Examples of a 1-cycle and a 2-cycle on a sphere.
 (b) Two 1-cycles representing the two equivalence classes in $H_1(\mathbb{X})$ for a torus. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

The ranks b_k of the homology groups $H_k(\mathbb{X})$ were named *Betti numbers* by Poincaré after Enrico Betti (1823–1892). b_0 has a simple geometric interpretation as the number of connected components of \mathbb{X} . b_1 is the number of non-equivalent closed curves, which is twice the genus of the surface in case \mathbb{X} is an orientable surface, and the genus minus one in case \mathbb{X} is a non-orientable surface. b_2 can be interpreted as the number of 3D spaces enclosed by the surface. It is one if \mathbb{X} is orientable since in this case \mathbb{X} separates the 3D space into two subspaces, according to Jordan-Brouwer's theorem (also known as the generalized Jordan curve theorem; see, for example, [MUN 84, HAT 02]). It is zero otherwise, since a non-orientable surface is not the boundary of any 3D object. The alternating sum of the Betti numbers $b_0 - b_1 + b_2$ is called the *Euler characteristic* $\chi(\mathbb{X})$ of the surface. Hence, the Euler characteristic of a connected orientable surface is $2 - 2g$ with g being the genus of the surface. The Euler characteristic of a non-orientable surface is $2 - g$.

For non-orientable surfaces, homology is also interested in the k -cycles that are not the boundary of a $k + 1$ -dimensional chain, but become a boundary if taken a certain number λ_k times. Numbers λ_k are called *torsion coefficients*. They enable us to classify non-orientable k -cycles. For example, a Klein bottle (see Figure 2.1 (c)) has a torsion coefficient λ_1 equal to 2.

The Betti numbers b_k and the torsion coefficients λ_k give the complete homology information about a surface described as a simplicial complex. Any set of $b_k + \lambda_k$ representative k -cycles (one for each equivalent class of $H_k(\mathbb{X})$) is called a set of **k -generators** of the surface. 1-Generators are of particular interest since they allow the flattening of a surface to a polygon. Such a polygon is called the *fundamental polygon* or *canonical polygonal schema* of the surface and only depends on its orientability and its genus. The corresponding 1-generators are called a *canonical set of generators* or a *canonical homology basis*. Such generators are simple (i.e. without self-intersection) closed curves that meet in a single point called the *basepoint* of the set (see Figure 2.5). If each curve is given an arbitrary direction and a letter a_i or b_i , the sequence of arrivals of the curves to the basepoint defines the *word* of the fundamental polygon. More specifically, we have:

– any orientable surface of genus $g \geq 1$ can be flattened to a $4g$ -gon, whose word can be written as $a_1 b_1 a_1^{-1} b_1^{-1} \dots a_g b_g a_g^{-1} b_g^{-1}$;

- any non-orientable surface of genus $g \geq 1$ can be flattened to a $2(g - 1)$ -gon, whose word can be written as $a_1 a_1 \dots a_{g-1} a_{g-1}$;
- a sphere can be represented by a two-sided polygon, whose word can be written as $a_1 a_1^{-1}$ (degenerate case).

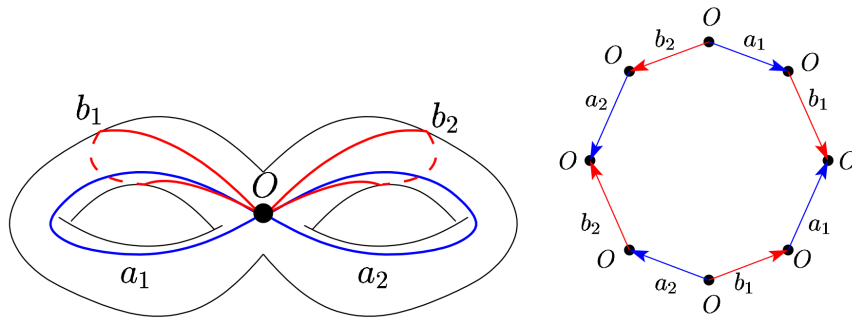


Figure 2.5. Canonical set of generators on a double torus, meeting in a basepoint O (left) and the corresponding fundamental polygon (right). For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

2.2. Computation of global topological features

In this section, we review algorithms to compute global topological features, as defined above, on a polyhedral surface. Except when stated, our input is a simplicial complex.

2.2.1. Connected components and genus

The connected components of a polyhedral surface can easily be computed in linear time (w.r.t. the number of vertices or polyhedra of the surface). Starting from any vertex v on the surface, a depth-first search or a breadth-first search on the graph made up of the vertices and the edges of the surface enables us to retrieve the connected component containing v . Consequently, a procedure to compute all connected components loops over the vertices of the surface, initially labeled as *not discovered*. For each *not discovered* vertex v , a new connected component is created. A depth-first search or a breadth-first search is then started from v , labeling each vertex

reachable from v (including v) as *discovered* and belonging to the same connected component as v . Another method to compute the connected components of the surface uses a disjoint-set data structure [COR 09]. Such a data structure operates on disjoint dynamic sets that are identified through a representative element. In our case, the sets contain the vertices of the polyhedral surface. A disjoint-set data structure typically defines a $\text{Find}(v)$ procedure, which returns the representative of the set containing the element v , as well as a $\text{Union}(v, w)$ procedure, which merges the sets containing the elements v and w . At the beginning, one set is created per vertex, containing only this vertex. Computing the connected components boils then down to browse the edges of the surface. For each edge vw , we only need to find the sets containing vertices v and w respectively and merge them if they differ, in order to create one set per connected component. Both the $\text{Find}(v)$ and $\text{Union}(v, w)$ procedures can run in almost constant amortized time if the sets are implemented using forests and procedures are carefully implemented [COR 09], making the whole computation run in linear time.

The genus of a connected polyhedral surface \mathbb{X} is easy to compute using its Euler characteristic $\chi(\mathbb{X})$. Remember from the previous section that the Euler characteristic of a connected surface is linked with its genus: $\chi(\mathbb{X}) = 2 - 2g$ if \mathbb{X} is orientable, $\chi(\mathbb{X}) = 2 - g$ otherwise. Moreover, *Euler's formula* states that the Euler characteristic of a polyhedron can be computed as the alternating sum of its numbers of elements of increasing dimensions: $\chi(\mathbb{X}) = V - E + F$ with V , E and F being the number of vertices, edges and faces respectively. We can therefore compute g from V , E and F . Note that if the surface is not connected, the Euler characteristics of its connected components sum up. For example, if \mathbb{X} is made up of two connected components \mathbb{X}_1 and \mathbb{X}_2 , then $\chi(\mathbb{X}) = \chi(\mathbb{X}_1) + \chi(\mathbb{X}_2)$.

2.2.2. Homology groups

The classical way to compute the homology groups of a simplicial complex is through the reduction of its incidence matrices to a canonical form called the *Smith normal form* (SNF) [DUM 03]. *Incidence matrices* are integer matrices which encode which simplices of the simplicial complex are boundary of a higher-dimensional simplex. Let us order the simplices of any dimension k of the simplicial complex arbitrarily $\sigma_0^k, \dots, \sigma_l^k$, then the incidence matrix I^k

has its element in row $i + 1$ and column $j + 1$ equal to 0 if σ_j^{k-1} is not in the boundary of σ_i^k , 1 if σ_j^{k-1} is in the boundary of σ_i^k and -1 if $-\sigma_j^{k-1}$ (the same simplex but with opposite orientation) is in the boundary of σ_i^k . Note that, as stated in section 2.1.2, all the simplices of the complex need to be arbitrarily oriented. For example, in the case of the simplicial complex shown in Figure 2.6, the incidence matrices I^1 and I^2 are respectively:

$$I^1 = \begin{pmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

$$I^2 = (1 \quad -1 \quad 1 \quad 0 \quad 0)$$

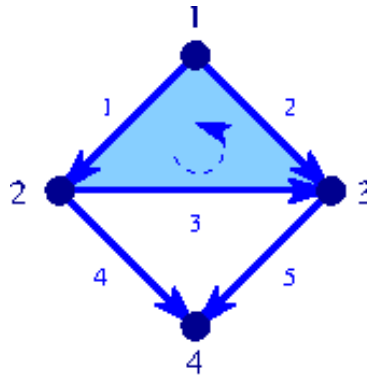


Figure 2.6. Toy simplicial complex example for the Smith normal form computation. Orientations of each 1-simplex (i.e. edge) and of the 2-simplex (i.e. triangle) are shown with arrows. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

An incidence matrix I^k is usually huge since its size is the number of simplices of dimension $k - 1$ times the number of simplices of dimension k . Its SNF is a diagonal matrix D^k with only a limited number of non-zero

diagonal elements, defined as $D^k = S^k I^k T^k$ with S^k and T^k being invertible square matrices. In the example of Figure 2.6, we have:

$$D^1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$D^2 = (1 \quad 0 \quad 0 \quad 0 \quad 0)$$

The link with the homology groups is the following:

- the number of zero rows of D^k is equal to the number of k -cycles;
- the number of non-zero columns of D^{k+1} is equal to the number of k -boundaries.

Thus, the Betti number b_k is simply the subtraction between these two numbers. In the case of the simplicial complex shown in Figure 2.6, we thus have $b_1 = 2 - 1 = 1$. In practice however, the reduction of an incidence matrix to its SNF is very time-consuming. Currently best-known algorithms, such as [DUM 03], have super-cubical complexity. This makes this method applicable only for simplicial complexes of small size.

In order to compute the homological information for simplicial complexes with a large number of simplices, a popular approach is to *reduce* the complex to a complex with significantly less simplices while preserving the homology. Discrete Morse theory, introduced by Forman [FOR 98], can be used for this purpose. *Morse theory*, named after its originator Marston Morse (1892–1977), is a mathematical tool to study the topology of a manifold \mathbb{X} with differentiable functions f . More specifically, f being smooth, the set of points of \mathbb{X} which share the same value for f , called a level set, is usually a set of loops on \mathbb{X} . Degenerate cases occur when there is a change in the topology of the level set, for example, two loops merging into one. They correspond to points for which the gradient of f vanishes. These points are called *critical points* of f on \mathbb{X} . For most functions, called *Morse functions*, critical points are isolated and the corresponding level set either is reduced to a single point or comprises several loops connecting in the critical point. In the first case, the point is a local

minimum or maximum for f while in the second case, it is called a saddle point (see Figure 2.7). The *index* of a critical point x is the number of negative eigenvalues of the Hessian matrix, i.e. the matrix of second derivatives of f at x . On a 2-manifold, this index is equal to 0 for a local minimum, 1 for a saddle point and 2 for a local maximum. All these notions and related results have been extended to the discrete case by Robin Forman in 1998 [FOR 98]. In particular, several inequalities relate the number n_i of critical points of index i and the Betti numbers. Let k be the dimension of the manifold \mathbb{X} ($k = 2$ in our case). We have:

$$\chi(\mathbb{X}) = b_0 - b_1 + \cdots \pm b_k = n_0 - n_1 + \cdots \pm n_k \quad [2.1]$$

$$\forall i \in [0, k], n_i \geq b_i \quad [2.2]$$

$$\forall i \in [0, k], n_i - n_{i-1} + \cdots \pm n_0 \geq b_i - b_{i-1} + \cdots \pm b_0 \quad [2.3]$$

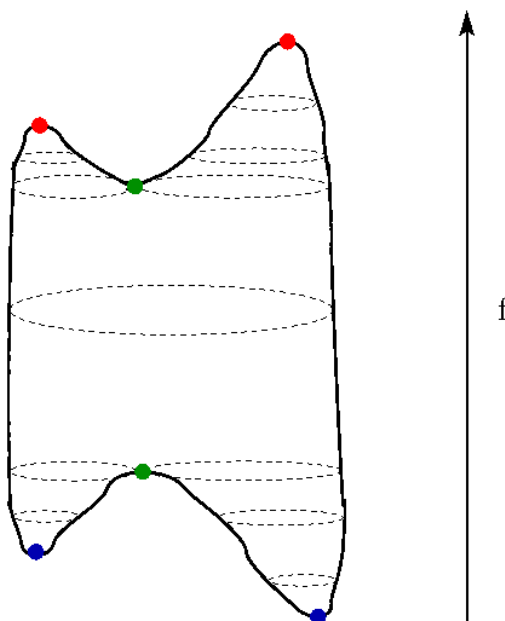


Figure 2.7. Morse theory: 2D example with a height function f . Critical points are shown in blue (local minima), red (local maxima) and green (saddle points). Some level sets are shown in dashed lines. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

Equations [2.2] and [2.3] are called *weak Morse inequalities* and *strong Morse inequalities* respectively. They can be used to efficiently compute the Betti numbers, since only the critical points of a Morse function f need to be computed. The Morse function should be carefully chosen so that its number of critical points is low with respect to the size of the simplicial complex. However, this number can still be high. Homological discrete vector fields have recently been introduced to overcome this problem [GON 17].

Finally, a third category of methods uses the theory of *constructive homology* [SER 99] to compute the homology of a simplicial complex. For example, Boltcheva *et al.* [BOL 11] decompose the input simplicial complex into so-called *Manifold-Connected* components [HUI 07], whose homology is then independently computed. This is done using the reduction of the corresponding incidence matrices to their SNF, which is fast since each component is small. The homology of the input complex is computed by iteratively calculating the homology of subcomplexes $A \cup B$ from the homology of A , B and $A \cap B$, starting from the Manifold-Connected components.

Methods have also been devised to compute homology groups in special cases. Although they lose generality, these methods are usually faster or more robust than previously described methods. An example is the recursive algorithm proposed by Gonzalez-Lorenzo *et al.* [GON 16b] to retrieve Betti numbers of a cubical complex, which is a specific case of a topological space with cubic cells. This method runs in linear time thanks to the regular structure of the complex.

2.3. Combining geometric and topological features

When dealing with topology as presented in the previous section, the geometry of the objects at stake is not taken into account. For example, if we consider a discrete object made up of *cells* (triangles, simplexes, cubes, voxels, etc.), the shape of these cells, their size or their position are not considered. This is coherent with the topological way of thinking, which considers spaces up to deformations. However, it is in practice often relevant to combine geometric and topological information. For example, once we know a given surface has g holes, we may want to recover the *size* of these holes. If its Betti numbers and torsion coefficients are known, it may also be

useful to compute a set of k -generators with good properties, for example, shortest ones.

This section reviews recent approaches proposed to compute geometric information related to topological features. We first introduce the concept of *persistent homology* (section 2.3.1), which filters topological features with respect to the size of associated geometric features. We then explain how the Morse theory presented in the previous section can be used to define two interesting tools giving global information about the surface with respect to a function, namely the *Reeb graph* and the *Morse–Smale complex* (section 2.3.2). The computation of locally shortest homology generators and other closed curves is reviewed in section 2.3.3. Finally, section 2.3.4 describes in detail a recently developed approach taking advantage of persistent homology to extract two independent measures for the homology groups of binary volumes.

2.3.1. Persistent homology

In this section, we introduce the concept of *persistent homology*. Although it can be applied to any simplicial complex, we take as example the case of *cubical complexes*. Note that some elements of this section are derived from [KAC 04]. The reader can refer to it for an in-depth understanding of these concepts.

An *elementary interval* is an interval of the form $[k, k + 1]$ or a degenerate interval $[k, k]$, where $k \in \mathbb{Z}$. An *elementary cube* in \mathbb{R}^n is the Cartesian product of n elementary intervals. The number of non-degenerate intervals in this product is its *dimension*. An elementary cube of dimension q is called *q-cube*.

Considering two elementary cubes σ and τ , then σ is a *face* of τ if $\sigma \subset \tau$. It is a *primary face* if the difference of their dimensions is 1. In the same way, σ is a *coface* of τ if $\sigma \supset \tau$. A *cubical complex* is a set of elementary cubes with all of their faces. The *boundary* of an elementary cube is the collection of its primary faces.

A *chain complex* (C_*, d_*) is a sequence of groups C_0, C_1, \dots (called *chain groups*) and homomorphisms $d_1 : C_1 \rightarrow C_0, d_2 : C_2 \rightarrow C_1, \dots$ (called *differential* or *boundary operators*) such that $d_{q-1}d_q = 0, \forall q > 0$.

For example, the chain complex of a cubical complex is defined as follows:

- C_q is the free group generated by the q -cubes of the complex;
- d_q gives the “algebraic” boundary, which is the linear operator that maps every q -cube to the sum of its primary faces.

The elements of the chain group C_q , which are formal sums of q -cubes with coefficients in \mathbb{Z}_2 , are called q -chains. They can be appreciated as sets of cubes of the same dimension. In the more general case of simplicial complexes, q -chains are formal sums of q -simplices with coefficients in \mathbb{Z} .

A q -chain x is a *cycle* if $d_q(x) = 0$, and a *boundary* if $x = d_{q+1}(y)$ for some $(q + 1)$ -chain y . By the property $d_{q-1}d_q = 0$, every boundary is a cycle, but the reverse is not true: a cycle that does not bound contains a “hole”. The q -th homology group of the chain complex (C_*, d_*) contains q -dimensional “holes”: $H(C)_q = \ker(d_q)/\text{im}(d_{q+1})$. This set is a finitely generated group, so there is a “base” typically formed by the holes of the complex. In the case of cubical complexes, since our ring of coefficients is \mathbb{Z}_2 , this group is isomorphic to $\mathbb{Z}_2^{\beta_q}$ and β_q is the q -th *Betti number*.

Considering a binary volume, it is possible to define two different associated cubical complexes encoding the 6- or the 26-connectivity relation.

- Primal associated cubical complex (26-connectivity): each voxel $x = (x_1, x_2, x_3)$ generates the 3-cube $[x_1, x_1 + 1] \times [x_2, x_2 + 1] \times [x_3, x_3 + 1]$ and all its faces.

- Dual associated cubical complex (6-connectivity): first, for every voxel (in fact, 3-clique²) $x = (x_1, x_2, x_3)$ of the volume, the 0-cube $\sigma = [x_1] \times [x_2] \times [x_3]$ is added. Then, for every d -clique ($d < 3$) in the volume, a $(3 - d)$ -cube is added to the cubical complex such that its vertices are the voxels of the d -clique.

It is possible to define these cubical complexes for any dimension. Figure 2.8 presents both these complexes.

² The notion of *clique* needs to be adapted to this context: a d -clique is a maximum (in the sense of inclusion) set of voxels such that their intersection is a d -cube.

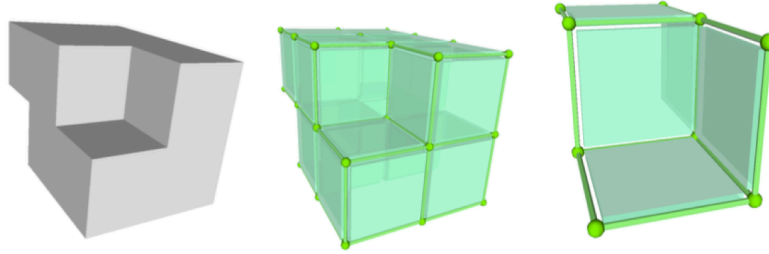


Figure 2.8. Left: a binary volume. Center: its primal associated cubical complex. Right: its dual cubical complex (courtesy [GON 16a]). For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

We now more formally introduce the notion of *persistent homology*, which initially comes from the concept of size functions introduced in 1990 by Patrizio Frosini [FRO 90]. A rigorous and complete description of persistent homology can be found in [EDE 08] and [EDE 17]. Here again, we use the case of cubical complexes as an example.

A *filtration* is a finite (or countable) sequence of nested (cubical) complexes $X_1 \subset X_2 \subset \dots \subset X_n$. It is also possible to describe it by a function f on the final complex X_n , which assigns to each q -cube the first index at which it comes out in the complex. As it is a sequence of cubical complexes, a cube cannot appear strictly before its faces, so the function f must verify:

$$f(\sigma) \leq f(\tau), \forall \sigma \subset \tau \quad [2.4]$$

Consequently, a function $f : X \rightarrow \mathbb{R}$ defined over a cubical complex is a *filtration function* if its image is a finite (or countable) set and if it makes equation [2.4] true. Considering such a function, its filtration is the sequence $\mathcal{F}_f(X) = \{X_i\}_{i=0}^n$, where $a_0 < a_1 < \dots < a_n$ are the images of f and $X_i = f^{-1}([-\infty, a_i])$.

As shown in Figure 2.9, the homology groups of the complex can change as “time” passes. The *persistence diagram* [EDE 08, p. 3] records these evolutions: a q -hole being born in X_i and vanishing in X_j is plotted in the persistence diagram as the point (i, j) . This is also named a *P-interval* in [ZOM 05]. A homology generator of X_n , which never dies, is described by

the point (i, ∞) . Persistent homology can also be represented in terms of *barcodes* [GHR 08], where each point (i, j) is visualized as an interval in the real line.

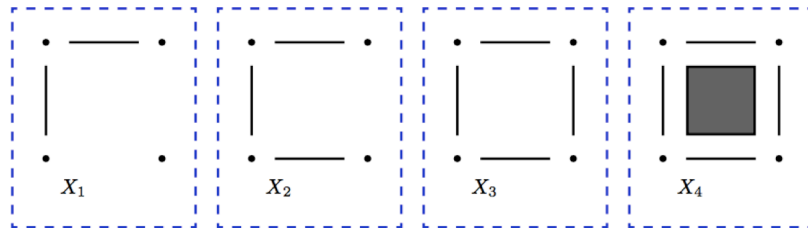


Figure 2.9. A filtration. X_1 : there are two 0-holes (connected components). X_2 : one 0-hole dies. X_3 : a 1-hole is born. X_4 : that 1-hole dies (courtesy [GON 16a])

Research dealing with the efficient computation of persistent homology has been considerable over the last decade. In addition to its theoretical aspects, persistence is increasingly used in application fields such as image processing, shape analysis, genetics, music, linguistics, neurosciences and robot navigation. In such fields, the “time” function is defined using non-topological information such as colors and geometry. More globally, the main tool involved in *topological data analysis* (TDA), a recent domain aiming to analyze high-dimensional datasets using techniques from topology, is persistent [EPS 11]. The reader can refer to [FER 16] and Chapter 3 for an overview of application fields of persistent homology. An example in which the “time” function is geometry-based is also presented in section 2.3.4, in order to compute the size of the holes of an object.

2.3.2. Reeb graph and Morse–Smale complex

We have seen in section 2.2.2 that a powerful tool to compute topological features is the Morse theory. In this theory, the critical points of a smooth function f on the manifold are used to compute the Betti numbers. We can go further and **connect** these critical points, according to f as well as to the surface geometry.

Let \mathbb{X} be our input surface and f be a smooth Morse function defined over \mathbb{X} . Connecting each critical point of f to its “neighbors” generates a

topological skeleton of the surface (see Figure 2.10 (a)). This skeleton is a graph named after the mathematician Georges Reeb, who proposed it in 1946 [REE 46]. Formally, the *Reeb graph* of f is defined as the quotient space \mathbb{X}/\sim , with \sim being the equivalence relation on \mathbb{X} such that $x_1 \sim x_2 \iff f(x_1) = f(x_2)$, and x_1 and x_2 belong to the same connected component of $f^{-1}(f(x_1))$. This way, the nodes of a Reeb graph correspond to the critical points of f , and its edges are made by sweeping the level sets of f and shrinking each connected component into a line. The definition of a Reeb graph is valid on any topological space. In the case of a simply connected Euclidean space such as a surface, the term *contour tree* is also used.

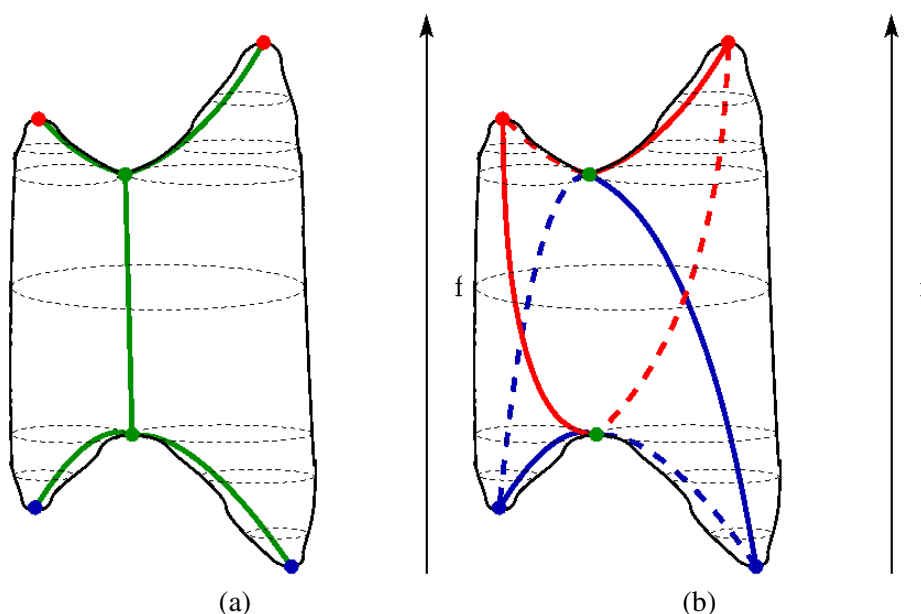


Figure 2.10. (a) *Reeb graph* (in green) of the function f for the example shown in Figure 2.7. (b) *Morse–Smale complex* of the function f . The blue curves are the descending manifolds of saddle points, while the red curves are their ascending manifolds. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

To the best of our knowledge, the Reeb graph has first been used in the visual computing community in 1991 by Yoshihisa Shinagawa *et al.* in order to reconstruct a surface from cross sections [SHI 91]. Since then it has been a popular tool for shape analysis and matching, mesh segmentation, surface

parameterization, computer animation, etc. (see [BIA 08] for more details about Reeb graphs and their extensions).

Beyond the contraction of the surface to a topological skeleton, the critical points of a Morse function f can be used to **decompose** the shape into cells. To do so, we are interested in the *integral lines* in f , i.e. the parametric curves $\gamma : [0, +\infty) \rightarrow \mathbb{X}$ verifying the condition:

$$\frac{\partial \gamma}{\partial t}(t) = \nabla f(\gamma(t)).$$

This condition states that at any of its points the velocity of the curve matches the gradient of f . Note that any integral line on \mathbb{X} ends at a critical point of f , where its velocity drops to zero, since critical points are by definition the points where the gradient of f vanishes. Let x be a critical point of f . The set of integral lines in f converging to x is a cell called the *descending manifold* of x . Similarly, the set of integral lines in $-f$ converging to x is called the *ascending manifold* of x . The dimension of such manifolds depends on the index i of the critical point. The descending manifold of a local minimum ($i = 0$) is the point itself, while its ascending manifold is a i -cell, i.e. a continuous piece of surface. Conversely, the descending manifold of a local maximum ($i = 2$) is a 2-cell while its ascending manifold is reduced to the point. Both the descending and the ascending manifolds of a saddle point ($i = 1$) are 1-cells, i.e. curves. Altogether, the descending manifolds decompose \mathbb{X} into a cell complex (a set of cells such that the boundary of each k -cell is the union of $(k - 1)$ -cells) called the *descending Morse complex*. Similarly, the ascending manifolds define the *ascending Morse complex*. In case any descending manifold intersects any ascending manifold only transversally, f is said to be a *Morse–Smale* function and the intersection of the descending Morse complex and the ascending Morse complex is called the *Morse–Smale complex* of f .

A Morse–Smale complex on a surface is a collection of quadrangular 2-cells, each of them having two saddle points, one local minimum and one local maximum as vertices, such that the boundary of a 2-cell is made up of curves joining either a local minimum and a saddle point or a local maximum and a saddle point (never two saddle points). Figure 2.10(b) gives a Morse–Smale complex for the surface and function shown in Figure 2.7. The surface is decomposed into four 2-cells, each of them having the two saddle

points, one maximum and one minimum as vertices. This is a general property: a Morse–Smale complex decomposes the surface into quadrangular cells lying on the critical points of the Morse–Smale function.

Many methods have been devised to efficiently compute Morse–Smale complexes on discrete surfaces (see [FLO 15] for a survey). One of their main interests lies in their simplification, for both topological and geometric applications. As explained in section 2.2.2, Morse functions can have many critical points on a surface, which can lead to time-consuming algorithms to compute the homology of the surface. Simplifying the Morse(–Smale) complex enables us to remove unnecessary critical points in an ordered manner, collapsing a maximum-saddle pair into a local maximum or a minimum-saddle pair into a local minimum. This leads to more efficient homology, or even persistent homology [MIS 13], computation algorithms. From a geometric point of view, Morse–Smale complexes have been widely used for vector field analysis and simplification (see, for example, [GYU 06]). In this case, the Morse–Smale function f is chosen so that its gradient field matches the vector field. Another application of Morse–Smale complexes is the hierarchical segmentation of the surface. This has proven especially useful in structural biology [NAT 06] and other fields.

2.3.3. Homology generators

Computing a set of 1-generators of a 2D discrete manifold is interesting, especially when the topology of the manifold is complicated, since it gives a visual representation of this topology. In particular, when the manifold is orientable, the set of generators boils down to a set of $2g$ representative cycles of the homology group $H_1(\mathbb{X})$, with g being the genus of the surface. Since there are two generators per topological “hole” in the surface – one around the tunnel and another around its corresponding handle (see Figure 2.11 (a))– this set allows us to **locate** tunnels and handles. Unfortunately, generators can be unnecessarily long and winding. Finding **shortest** generators is thus essential to give information about the **size** of topological holes.

In the literature, three main problems have been addressed:

- 1) *free homotopy*, which is interested in finding shortest cycles that can be continuously deformed from given homology generators (Figure 2.11);

2) *homotopy with basepoint*, in which all shortest generators share a common point and otherwise do not intersect (Figure 2.5);

3) *tightening* or *localization*, which aims to find a shortest generator given a single input generator or a homology class.

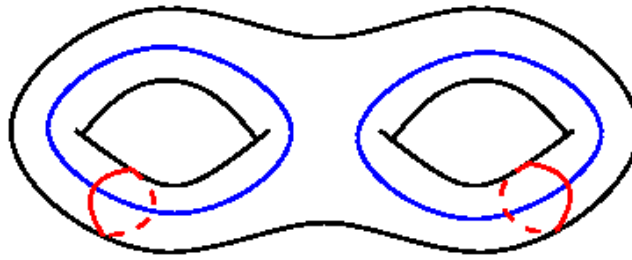


Figure 2.11. Set of 1-generators of $H_1(\mathbb{X})$ for a double torus (free homotopy). Generators corresponding to tunnels and handles are shown in blue and red respectively. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

In the first two cases, the length minimized is usually the total length of the generators. Note that both terms, loops and cycles, are used in the literature to describe 1-generators. The main difference is that a loop contains a distinguished vertex, called its *basepoint*, whereas a cycle does not have a repeated vertex [CAB 11]. Homotopy with basepoint is of particular interest since the corresponding generators, also called a *system of loops* in this case [COL 05], define the fundamental polygon of the surface (see section 2.1.2), which can be useful for parameterization or texture mapping. Most methods proposed to solve these three problems work on *combinatorial surfaces*, i.e. compute cycles on the edges and vertices of the discrete surface only, with positive weights (e.g. their length) associated with the edges. This is because exact length computation on piecewise-linear surfaces (i.e. allowing the cycle to cross the interior of planar faces) is technically more difficult, square roots being involved [COL 17].

One of the most effective algorithms to date to generate shortest free homotopy generators of a discrete 2-manifold is the one of Erickson and Whittlesey [ERI 05]. The generators are computed in $O(n^2 \log n + n^2 g + n g^3)$ time, with n being the number of cells (e.g. vertices or faces) of the surface and g being its genus. In addition, this algorithm

allows the computation of a shortest system of loops in $O(n^2 \log n)$ time. This improves upon a result by Colin de Verdière and Lazarus [COL 05]. In the more general case of simplicial complexes such as point clouds, Dey *et al.* proposed an algorithm that runs in polynomial time [DEY 10]. The problem of shortest free homotopy generators has also been studied for higher-dimensional manifolds. Chen and Freeman's algorithm [CHE 10] computes a smallest set of generators of $H_k(\mathbb{X})$ with a $O(b_k n^3 \log^2 n)$ complexity, with b_k being the rank of $H_k(\mathbb{X})$, i.e. the k -th Betti number of the manifold. In the same work, they also propose to use the length of these generators to measure 1-holes.

Dey *et al.* classify between handle and tunnel generators (see Figure 2.11) by looking at the first homology groups of the inside and outside volumes of \mathbb{R}^3 separated by the surface \mathbb{X} [DEY 08] (note that some generators can be neither handle nor tunnel ones). They also use persistent homology to compute such cycles. A more effective algorithm has been proposed later for the same purpose, using Reeb graphs this time [DEY 13]. Zomorodian and Carlsson's theory of *localized homology* [ZOM 08] is more general and aims to localize, from a geometric point of view, topological attributes of any dimension in any topological space.

Several methods exist to find a shortest generator in a given homology class. In case the generator is simple (no self-intersection), the approach of Colin de Verdière and Lazarus [COL 05] can be applied. Otherwise, the output-sensitive algorithm of Colin de Verdière and Erickson [COL 10] can be applied as soon as $g \geq 2$. The result is found in $O(gnk \log(nk))$ time, with k being the size of the resulting cycle. Other types of shortest closed curves on the surface have also been studied. Among them are *non-separating* cycles, which do not split the surface in two connected components, and *non-disk-bounding* cycles, which do not surround a topological disk on the surface. As a counterexample, the cycle in Figure 2.12 is separating while the red cycle in Figure 2.4(a) is disk-bounding (see [COL 17] for a review about such shortest curves). Finally, a *constriction* is defined as a locally shortest closed curve [HÉT 03]. It can be computed using persistent homology [FEN 13] and have a broader definition since it can be separating (Figure 2.12).

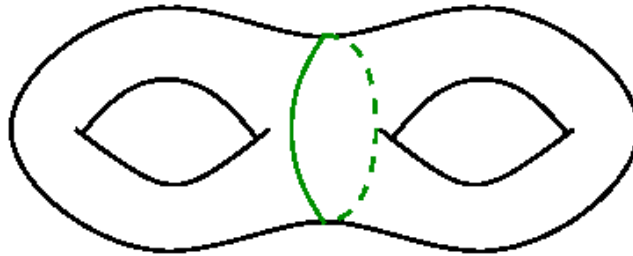


Figure 2.12. A constriction (in green) is a locally shortest cycle, in the sense of the Hausdorff distance

2.3.4. Measuring holes

We finish this chapter with a section dedicated to a recent approach developed by Gonzalez-Lorenzo *et al.* [GON 16a]. The main idea is that by taking a filtration based on the distance function, it is possible to link geometry and homology. The authors define two geometric measures in order to enrich the Betti numbers of a binary volume. These quantities are uniquely defined up to a choice of adjacency relation and distance. They are computed through a distance transform and using persistent homology, so it has matrix multiplication complexity over the number of voxels in the bounding box of the volume. These measures can be considered as pairs of numbers associated with each homology generator and also visualized as 3D balls on the volume.

One of the motivations of this work is related to the size of holes of a 3D shape. The objects in Figure 2.13 have the same Betti numbers ($\beta_0 = \beta_1 = 1$). They are equal from a homological point of view, but they look very different: the hole on the right is *bigger* than the hole on the left. The purpose of Gonzalez-Lorenzo *et al.* is to precisely discriminate shapes using measures on holes.

The hole on the left is *smaller* than the hole on the right because we need to add a smaller area to the object to fill the hole. Figure 2.14 shows how it is possible to fill the 1-holes by adding a *patch* (in red).

This leads to another idea: it is possible to erase holes also by removing a part of the object. Indeed, the hole on the left is *thicker* than the hole on the

right because we need to remove a bigger area to break the hole. Figure 2.15 shows how it is possible to break the 1-holes by erasing a *patch* (in blue). Thus, there is second measure for the holes, related to their breakability.

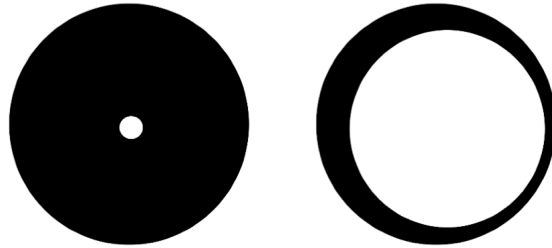


Figure 2.13. *Two objects with isomorphic homology groups*

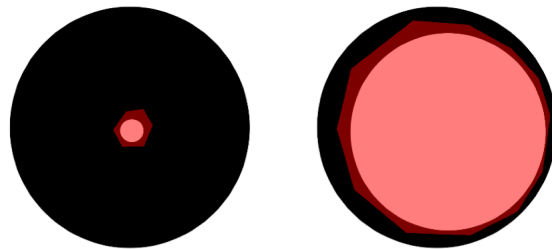


Figure 2.14. *By adding a patch (in red), the holes disappear. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip*

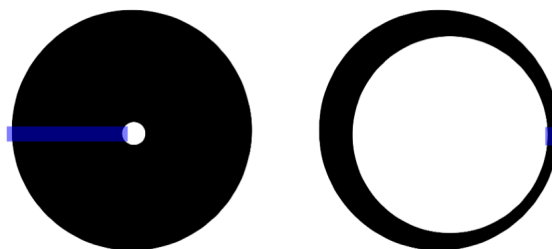


Figure 2.15. *By removing a part (in blue), the holes disappear. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip*

The two measures

In this work of Gonzalez-Lorenzo *et al.*, the considered filtrations are based on a function defined over a cubical complex. However, the reader can note that most of the time in the persistent homology literature, e.g., Vietoris-Rips or Alpha complexes are used instead.

The two measures defined by Gonzalez-Lorenzo *et al.* can be apprehended intuitively: considering two objects that are repeatedly eroded and dilated, holes are disappearing or created and changes are noticeable at different steps. These successive erosions/dilatations can be encoded by persistent homology, using a filtration based on the *signed distance transform*, and the measures are resulting from the “time” (i.e. step of the filtration) it takes to break or to fill a hole.

The idea to get a *measure* for homology classes has been developed by Chen and Freedman [CHE 10]. This measure is usually called *radius*, and it operates on simplicial complexes. However, computing the measure of a class is known to take $\mathcal{O}(n^4)$ time and there is no consideration for the geometry of the complex. On the contrary, the measures proposed by Gonzalez-Lorenzo *et al.* are only defined for discrete objects but can be computed faster and they have good geometric properties.

Let $O \subset \mathbb{Z}^d$ be a discrete object. Let fix $d = 3$ for simplicity, but the generalization to any dimension is direct. Let us consider K the cubical complex associated with O and constructed using the 6-connectivity relation.

The *distance transform* dt_O of O is the map that sends every voxel $x \in O$ to

$$dt_O(x) = d(x, O) = \min\{d(x, y) \mid y \notin O\}$$

where

$$d(x, y) = \sqrt{\sum_{i=1}^3 (x_i - y_i)^2}$$

is the Euclidean distance. Other distances such as the Manhattan distance (L_1), the chessboard distance (L_∞), distances based on chamfer masks [MON 68, BOR 84] or sequences of chamfer masks [MUK 00, NOR 09] could be considered.

The *signed distance transform* sdt_O of O is the map $sdt_O : \mathbb{Z}^3 \rightarrow \mathbb{R}$ defined as follows:

$$sdt_O(x) = \begin{cases} -dt_O(x) = -\min\{d(x, y) \mid y \notin O\} & \text{if } x \in O \\ dt_{\mathbb{Z}^3 \setminus O}(x) = \min\{d(x, y) \mid y \in O\} & \text{if } x \notin O \end{cases}$$

Let us point out some simple properties about the sublevel sets $L_t^-(sdt_O) := sdt_O^{-1}([-\infty, t])$ of the signed distance transform:

- 1) $O = sdt_O^{-1}([-\infty, 0])$.
- 2) $\mathbb{Z}^3 = sdt_O^{-1}([-\infty, \infty])$.
- 3) $sdt_O^{-1}([-\infty, a]) \subset sdt_O^{-1}([-\infty, b])$ whenever $a < b$.

Figure 2.16 shows five sublevel sets at different values. Observe that the sequence of objects $(sdt_O^{-1}([-\infty, t]))_{t=-\infty}^0$ looks like an erosion of O , while $(sdt_O^{-1}([-\infty, t]))_{t=0}^\infty$ seems to be a dilation.

Gonzalez-Lorenzo *et al.* define the filtration associated with the signed distance transform. A simple formulation of this filtration is

$$F = (K[L_t^-(sdt_O)])_{t \in \mathbb{R}}$$

where $K[Y]$ denotes the primal or the dual associated cubical complex of Y .

$PD_q(F) \subset \mathbb{R}^2$ denotes the persistence diagram in dimension q of this filtration. Let us denote $TB_q = \{(x, y) \in PD_q(F) \mid x < 0, y > 0\}$. It is clear that TB_q contains $\beta_q(K)$ pairs, i.e. there are as many pairs in TB_q as q -holes in O .

Let $O \subset \mathbb{Z}^3$ be a discrete object. Let us fix a distance function $d : \mathbb{Z}^3 \times \mathbb{Z}^3 \rightarrow \mathbb{R}$ and a connectivity relation. Let $q \geq 0$:

- the *thickness* of the q -holes of O is the values $\{-x \mid (x, y) \in TB_q\}$;
- the *breadth* of the q -holes of O is the values $\{y \mid (x, y) \in TB_q\}$.

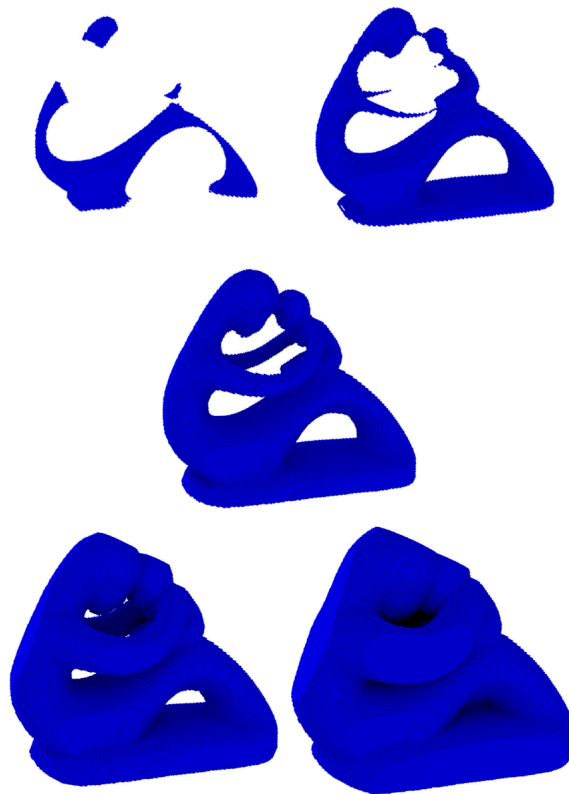


Figure 2.16. Sublevel sets of the signed distance transform at values -10 (top-left), -5 (top-right), 0 (middle), 5 (bottom-right) and 10 (bottom-right)

Observe that the thickness and the breadth of the holes appear in pairs. It is now possible to represent them as points in \mathbb{R}^2 in the *thickness–breadth diagram*, similar to the persistence diagrams. The authors call these points *thickness–breadth pairs*. The interpretation of the thickness–breadth diagram is similar to that of the persistence diagram: points close to the axes are holes with one small measure that may be originated by the presence of noise in the discrete object.

Thickness and breadth balls

One interesting asset of the two measures presented heretofore is that it is possible to “materialize” them by representing balls. This goes further than representing the thickness and the breadth of the holes as points in the diagram.

Each thickness–breadth pair (t, b) has an related pair of cubes (σ, τ) . Therefore,

- its *thickness ball* is the ball centered at the barycenter of σ with radius t ;
- its *breadth ball* is the ball centered at the barycenter of τ with radius b .

We can note that the thickness balls are contained inside the object, while the breadth balls are outside.

Thus, these balls allow the representation of the two measures directly on the object. Moreover, the breadth balls are in the center of the holes.

Visualizing holes as representatives for a set of homology generators is classically accepted. These representatives can be visually unpleasant. In order to formalize this aspect in a better way, some authors suggest that the best representatives are those which are minimum in terms of their length, area, volume, etc. [ERI 05, CHE 10, DEY 13].

Breadth balls are a natural and an alternative way to represent holes in terms of homology generators. Symmetrically, thickness balls materialize in some way the cohomology generators.

In the following, some examples of thickness–breadth diagrams and balls are presented³ in Figures 2.17, 2.18 and 2.19. Before the computation of the measures and balls, the meshes have been converted to binary volumes. The thickness balls are in red, while the breadth balls are in green. Only the balls of the 1-holes are displayed, since the other ones are less visually interesting. The thickness–breadth diagram shows the 0-holes (red circle), 1-holes (green triangles) and 2-holes (blue square). We can note that the thickness of the only connected components, which is ∞ , is represented as -1 .

³ Courtesy of the Aim@Shape repository.



Figure 2.17. *Fertility*: there are five, and not four 1-holes. The most fragile parts of the object are represented by the thickness balls in the arms and between the heads. The thickness ball in the base is more difficult to appreciate, as the hole formed by the legs does not have a tubular shape. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

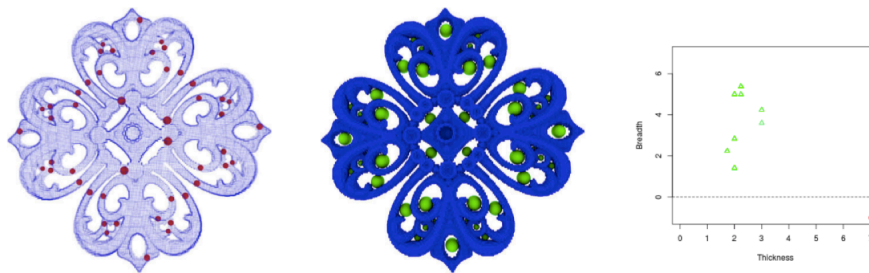


Figure 2.18. *Filigree*: the position of the thickness and the breadth balls respects quite well the symmetry of this object. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

This work developed by Gonzalez-Lorenzo *et al.* introduces a concise and rigorous geometric and topological information for discrete objects that extends the Betti numbers. These topological features are well suited for statistical analysis or for shape understanding and classification. However, many questions remain: the stability of breadth and thickness under small perturbations of the volume and the possible change of these values when different connectivity relations or distances are considered.

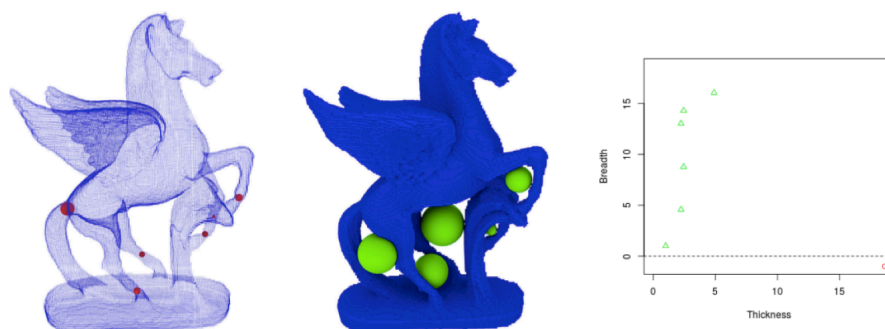


Figure 2.19. *Pegasus*: there are five significant holes and a small one near the right front paw (see its thickness ball). For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

2.4. Conclusion

We have seen in this chapter that topology defines global features about a shape, which do not depend on lengths or areas. These features are mathematically defined using the concept of *homology*. In case of a 2-manifold surface \mathbb{X} , they are:

- the number of *connected components* of the surface;
- its *genus* (number of “holes”);
- a set of closed loops around each hole and handle, called the *generators* of the homology group $H_1(\mathbb{X})$;
- in case the surface is not orientable, which means it is not possible to define an *interior* and *exterior* volume, *torsion coefficients* and their associated generators.

We have described algorithms to compute such features for orientable surfaces. We have also seen that geometric information can be used in conjunction with these features, either to make them more relevant for potential applications (e.g. compute shortest generators) or to sort them

according to their importance (e.g. the size of the holes using the concept of persistent homology). We should mention the existence of the Topological ToolKit (TTK) software [TIE 17], which provides the computation of most of the topological features mentioned here (and many more) for surfaces as well as other types of data.

In the next chapter, we will review application fields in which geometric and topological surface features are used.



Applications

3.1. Introduction

In this chapter, we show five scientific application contexts in which the concepts presented in the two previous chapters are used. They show that geometric and topological features are useful to understand a shape. Such features can help to either detect a specific area (sections 3.2, 3.4 and 3.5) or analyze the overall shape (sections 3.3 and 3.6).

3.2. Medicine: lines of curvature for polyp detection in virtual colonoscopy

In medical practice, the shape is a key parameter to recognize an anatomical structure and to differentiate between healthy or disease state. For example, in colonoscopy videos, gastroenterologists look for small abnormal extrusions of the colon wall, called polyps, which may be early indicators of colorectal cancer. The shape of the polyp can be defined according to some standard guidelines as for example the Paris classification [WOR 03]. In this classification, polyps are divided into morphological classes named “protruded”, “pedunculated”, “superficial”, “flat”, “depressed” or “excavated”.

In virtual colonoscopy (or CT colonography), the surface of the colon is reconstructed from a 3D CT scan image of the patient, and the endoscopic image is replaced by the visualization of the 3D surface mesh. It becomes then

possible to define differential parameters to extract geometric features in order to automatically detect polyps based on the Paris classification.

A first computer-aided detection application was described more than 15 years ago in [YOS 02]. The authors propose to compute the shape index s (see equation [1.12]) that locally defines the surface type as “cup” ($s \approx 0.0$), “rut” ($s \approx 0.25$), “saddle” ($s \approx 0.5$), “ridge” ($s \approx 0.75$) or “cap” ($s \approx 1.0$) and the “volumetric curvedness” c , which is based on the mean curvature and quantities the local flatness or sharpness of the surface. Polyp area candidates are then defined by $s \in [0.9, 1.0]$ (this means that they are cap shaped) and $c \in [1/12.5, 1/5]$ (with c values given in mm^{-1}). For each candidate, a region-growing process, where the values of s and c are relaxed, allows the segmentation of a significant portion of the polyp. The results are encouraging, but the false-positive rate remains high.

In fact, analyzing only scalar curvature values may be too limited to efficiently characterize the surface shape around the polyp. The idea is then to use the vector field given by the principal directions. In [ZHA 06], the authors propose to compute lines of curvature (see section 1.2.3) on the colon surface and detect some specific patterns. First, principal curvatures and directions are computed on the 3D surface mesh given by the segmentation of the colon in the 3D CT scan image. Then, lines of curvature are traced using a step-wise integration of the principal direction field from some seed points defined on the surface. If we superimpose these lines of curvature on the shaded representation of the colon surface, they greatly enhance the perception of the shape of polyps. Moreover, around the neck of the polyp, the shape is roughly cylindrical which means that the lines of curvature are (almost) closed, composed of points that are hyperbolic (i.e. with principal curvatures of opposite signs, see section 1.2.8). These two criteria allow us to automatically select a very small number of lines of curvature which are very characteristic of the polyp shape.

In a second paper [ZHA 08b], the authors extend the method by clustering lines of curvature in order to partition the colon surface (see Figure 3.1, upper row). The idea is that an area where lines of curvature are parallel has a consistent and rather simple shape. By computing the shape index s and the volumetric curvedness c for each cluster, it is then possible to detect polyp candidates with a rather good sensitivity and specificity (see Figure 3.1, bottom row).

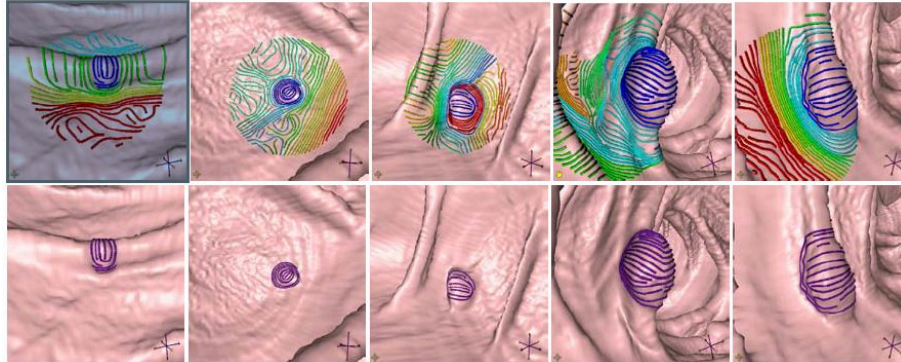


Figure 3.1. Upper row: partition of the colon surface by clustering lines of curvature. Clusters are indicated by different colors. Bottom row: automatic identification of the cluster that corresponds to a polyp. Figure reproduced from [ZHA 08b] by permission of the Eurographics Association. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

3.3. Paleo-anthropology: crest/ridge lines for shape analysis of human fossils

Human paleontology or paleo-anthropology is the science that aims to study the evolution of human species by analyzing the fossil remains found on an excavation site. By comparing these data with the ones of other excavation sites, it is possible to reconstruct the human evolution scheme.

For two decades, we have seen the emergence of computer-assisted paleo-anthropology [ZOL 98], also called virtual paleo-anthropology. It consists of acquiring a 3D image of the fossils, in general by using a medical or industrial CT scanner. It allows the paleontologist to visualize and analyze the very fragile fossil structures and to investigate their internal structures in three dimensions more easily than on real fossils.

In [DEA 96], the authors propose to study the transition between *Homo erectus* and *Homo sapiens* by analyzing the 3D shape of four anatomical lines: the “brow ridge” that is much more pronounced for *Homo erectus*, the “temporal line” to which the temporal muscle, one of the mastication muscle, is attached which is more salient for *Homo erectus*, the “coronal suture” that characterizes the profile of the sagittal keeling, and the “nuchal line” that emphasizes the so-called occipital bun, a protrusion at the back of the skull.

In [BOO 88], it is shown that these anatomical lines are related to the crest/ridge curves of the skull surface and can be defined by computing local maxima of curvatures. This set of crest/ridge curves can be compiled to define a wireframe template that characterizes the overall shape of the skull for medical [CUT 95] or anatomical applications [DEA 98].

In [SUB 02], the authors present an automatic method that allows us to visualize and analyze, in 3D, the evolution of the shape of the human skull. First, crest/ridge lines are automatically extracted from 3D CT scan images of the skull of a modern man and of a prehistoric man. Many of those lines then correspond to anatomical lines of the skull surface. Crest/ridge lines are then used as landmarks to automatically find corresponding points between the prehistoric and the modern skulls. A 3D transformation that superposes at best the two skulls is then computed based on couples of corresponding points. It makes then possible to visualize and analyze the 3D deformation of the skull shape between the prehistoric and the modern man. The method was applied to the skull of the “Tautavel Man”¹ (also known as Arago XXI), a *Homo heidelbergensis* fossil, dated about 450,000 years old. In particular, it allowed us to infer a reconstruction of the Tautavel Man’s face by deforming the corresponding face of the modern skull.

In [SUB 95], we can also find some results about the “Broken Hill” skull (also known as Kabwe 1), a *Homo rhodesiensis*² dated between roughly 300,000 and 125,000 years ago (see Figure 3.2). The 693 crest/ridge lines emphasize all the external and internal salient substructures of the skull and correspond to anatomical lines defined by anatomists. In particular, they clearly reveal some of the main features of the *Homo rhodesiensis* species as the great inflated brow-ridges that are especially prominent and prolonged to a greater extent at the lateral angles [WOO 21].

Note that the relationship between the anatomical lines and the computed crest/ridge lines of the skull was carefully analyzed in [THI 96c]. The conclusion is that we can get coherent results to compute a 3D deformation by using either of the two sets of lines.

1 http://en.wikipedia.org/wiki/Tautavel_Man.

2 http://en.wikipedia.org/wiki/Homo_rhodesiensis.

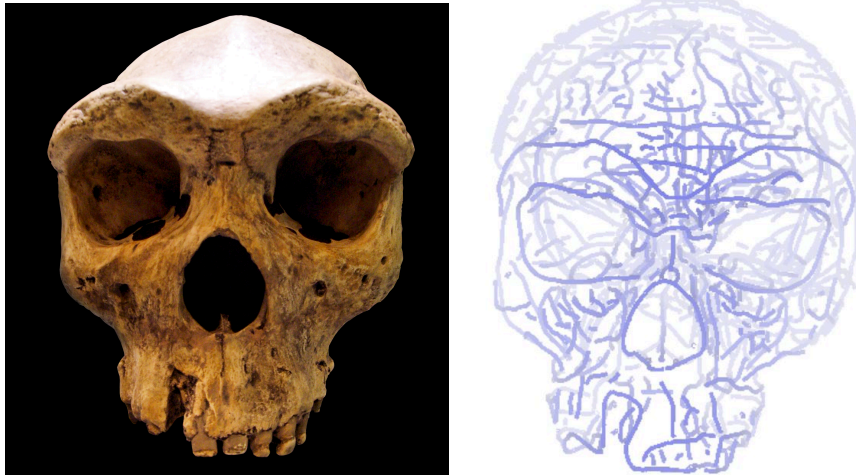


Figure 3.2. *Left: the Broken Hill prehistoric fossil skull³ is dated between roughly 300,000 and 125,000 years ago. Right: crest/ridge lines are automatically extracted from the 3D CT image of the fossil skull. They emphasize all the external and internal salient substructures of the skull. Some of them are characteristic of the *Homo rhodesiensis* species. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip*

More recently, in [SUB 09], the authors propose to use the method described in [YOS 08] to extract crest/ridge lines and analyze the fossil shape of two other anatomical structures, the Enamel Dentine Junction and the endocranium.

The enamel–dentine junction (EDJ) is the boundary surface between the enamel and the underlying dentin that forms the solid architecture of a tooth. The EDJ is the earliest feature to appear in dental development, well before the functional emergence of the tooth. Examination of the EDJ allows us to study the tooth crown morphology, even when the tooth has been modified by a strong attrition. The EDJ surface can be segmented in a μ CT scan image of a fossil tooth, where the enamel part is particularly visible. In Figure 3.3, left, we can see how crest/ridge lines emphasize the pattern of grooves and

³ [http://commons.wikimedia.org/wiki/File:Broken_Hill_Skull_\(Replica01\).jpg](http://commons.wikimedia.org/wiki/File:Broken_Hill_Skull_(Replica01).jpg) by Gerbil [CC BY-SA 3.0]

cusps on three molars of an *Australopithecus africanus*. This is very useful as it has been shown that the number of cusps and grooves characterizes hominid species [SKI 09] and differentiates tooth types [BRA 10].

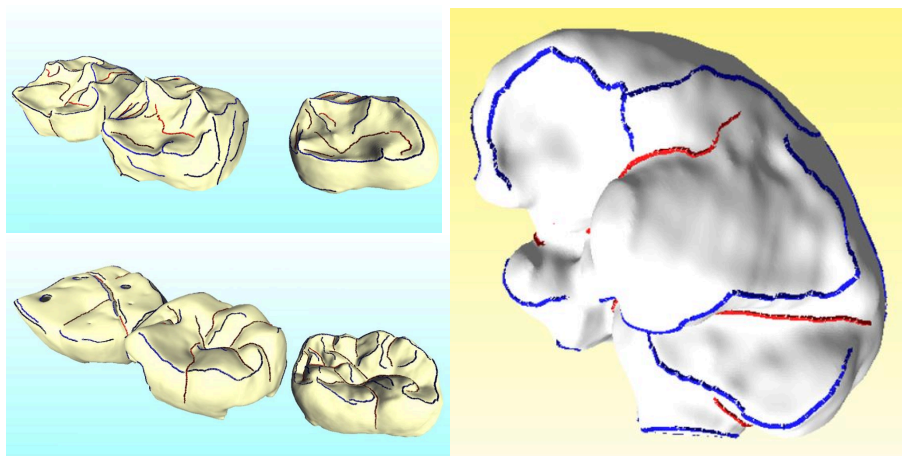


Figure 3.3. Crest/ridge lines extracted from two fossil structures of *Australopithecus africanus* dated about 2.1–2.5 million years old. Left: crest/ridge lines automatically computed on the enamel–dentin junction surface of the teeth of the fossil STS52⁴ Right: crest/ridge lines extracted from the endocranial surface (i.e. the internal surface of the skull) of the fossil STS5⁵ (also nick-named “Mrs. Ples”). For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

The endocast is the inside of the braincase and its surface is considered as providing a kind of replica of the brain surface with the impression of cerebral gyri and sulci or vessel patterns. The analysis of the endocast surface allows the anatomist to estimate the brain size, the lobe division that may be related to different brain functions or the vascularization of different parts of the brain. More generally, the endocast shape gives important information about the hominid species [BEA 16]. The surface of the endocast can be segmented in a CT scan image of a fossil skull by “filling” the brain case. In Figure 3.3, right, we can see how crest/ridge lines can delineate the endocast surface of an *Australopithecus africanus* emphasizing the parietal lobe or the cerebellum. Moreover, by distinguishing between concave (in red) and

⁴ [http://www.efossils.org/page/boneviewer/Australopithecus africanus/Sts 52.](http://www.efossils.org/page/boneviewer/Australopithecus+africanus/Sts+52)

⁵ [http://www.efossils.org/page/boneviewer/Australopithecus africanus/Sts 5.](http://www.efossils.org/page/boneviewer/Australopithecus+africanus/Sts+5)

convex (in blue)-shaped crest lines, we can approximate some of the main gyri and sulci [SUB 99b].

3.4. Geology: extraction of fracture lines on virtual outcrops

Among the approaches detailed in Chapter 1, the one in section 1.4.4 has been developed in the frame of an automatic detection of geological objects on 3D virtual outcrops [KUD 11, KUD 13]. The geological features to be extracted are the *fracture lines* and the *stratigraphic limits*. It can naturally be transposed as the detection of feature lines on a 3D surface mesh, for the areas with high positive values of the mean curvature (which correspond to strongly concave areas, intuitively where the surface has locally valley-like shapes).

Figure 3.4 shows input data before their processing. Starting from LIDAR acquisitions of an outcrop, the output cloud of points is triangulated.

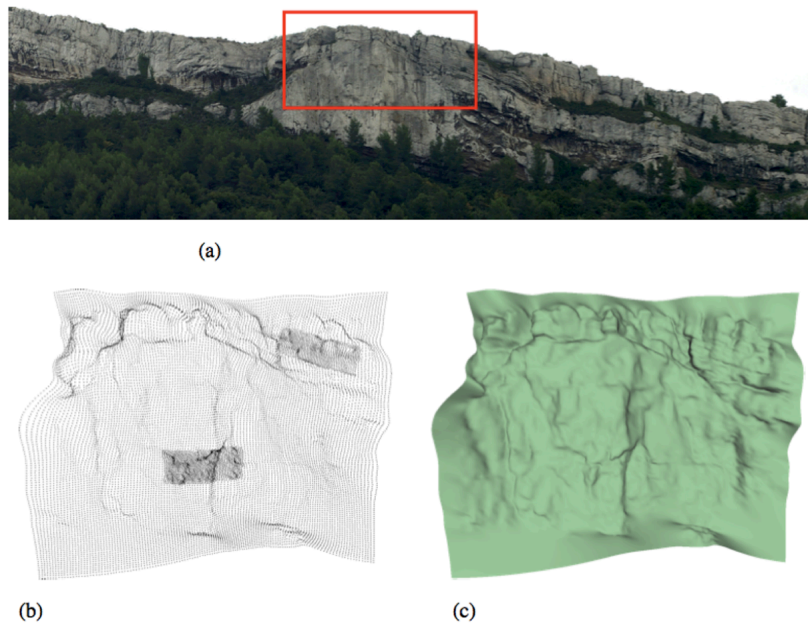


Figure 3.4. (a) Area of acquisition of the outcrop (La Marcouline, Cassis, France). (b) Cloud of points obtained using a LIDAR. (c) Surface mesh computed from the sparse 3D data points

In Figure 3.5, each vertex of the mesh is labeled according to the sign of its mean curvature to define a set of vertices (the area of interest). Then, using the skeletonization operator defined in section 1.4.4 (based on the homotopic thinning of regions lying on a surface mesh), the feature lines are extracted.

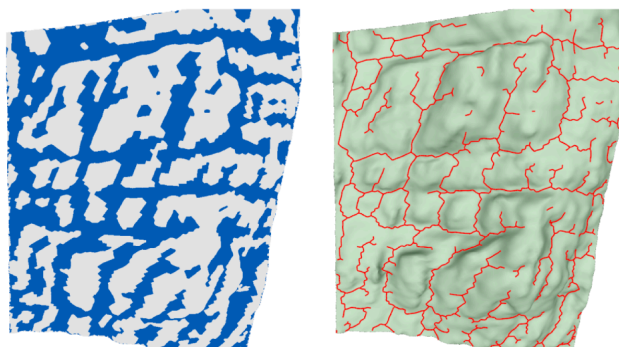


Figure 3.5. *On the left: areas of interest. On the right: extracted skeletons. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip*

Figure 3.6 shows the results obtained on a virtual outcrop using four different approaches: two classical approaches (a) and (b) and two approaches based on vertex labeling and thinning (c) and (d).

The methods (a) [DEC 03] and (b) [YOS 05] are based on the study of the variation of curvatures, which means derivatives of the third order are computed for each vertex. It implies a longer computing time, an instability due to the high-order derivative and a lack of connectivity of the branches of the feature lines (umbilic points and T-junctions are singularities that cut the lines).

The methods (c) [KUD 11] and (d) [KUD 13] are both based on the labeling of regions of interest (vertices with a positive mean curvature) and a thinning of these regions. In the approach (c), each region of interest (each connected component) is unfolded into a regular 2D grid. Then, a skeletonization is performed. It is a homotopic thinning preserving the topology of the branches. However, a distortion can appear because of the

parameterization step. In the approach (d), all the regions of interest are processed in the same pass, with no unfolding. A skeletonization operation is performed directly on the triangulated mesh, based on a classification of the vertices and a characterization of their topology. Thus, by removing only *simple vertices* (which do not modify the topology of the shape), the skeleton (and the feature lines) are extracted.

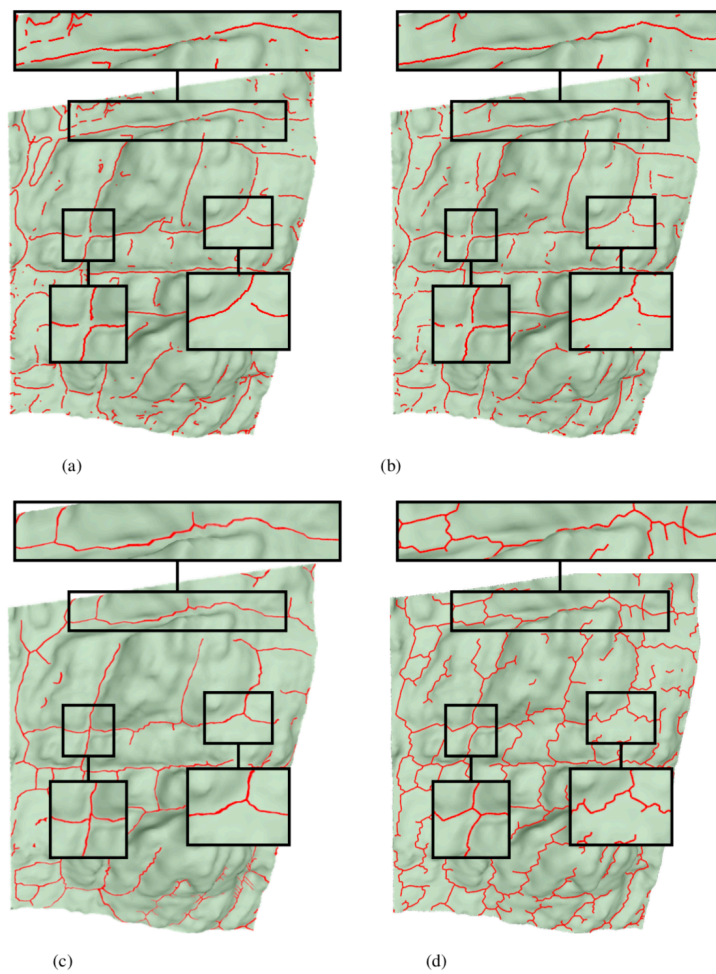


Figure 3.6. Comparison between two classical approaches (a) and (b) and two approaches based on vertex labeling and thinning (c) and (d)

3.5. Planetary science: detection of feature lines for the extraction of impact craters on asteroids and rocky planets

Astrophysicists have frequently used digital imaging approaches to characterize the properties of rocky celestial objects. This has led to the development of many 2D methods. Today, thanks to embedded systems, it is possible to exploit 3D models at high resolutions containing all information required for the detection of geomorphological features.

Most celestial bodies show impacts of collisions with asteroids and meteoroids. These traces are called craters. The possibility of identifying these craters and their characteristics (radius, depth and morphology) is the only method available to evaluate the age of different units at the surface of the body, which in turn allows us to constrain its conditions of formation (see [MAR 10, FAS 16]).

This example of application [MAR 19] aims to develop an automatic detection method for the craters lying on these 3D models. Mari *et al.* have elaborated two variants based on discrete curvature computation. The first method uses curvature estimations and vertices labeling to detect specific concave depressions on the surface. Concentric rings are then built around these depressions to find a circular rim using the curvature values of the points on the ring. This method has been tested on a 3D model of the asteroid Lutetia observed by the ROSETTA (ESA) space probe. The second method is based on the detection of closed circular areas of crater edges. The deepest point gives the contour of the area and then a circle is fitted.

Figure 3.7 presents the propagation process of the first approach, for one seed detected at the center of a predictable crater. A contour propagates starting from the seed to the associated rim (the vertices where the mean curvature H is negative). The contour is exclusively composed of edges of the initial triangle mesh.

Figure 3.8 shows the result of the first automatic detection method. This approach offers an effective detection rate of 85% and a false positive rate of 10% within a single pass (see Figure 3.8), which is comparable to automatic 2D approaches. Some of the larger rings are slightly off-centered as a result of an incorrect epicenter placement.

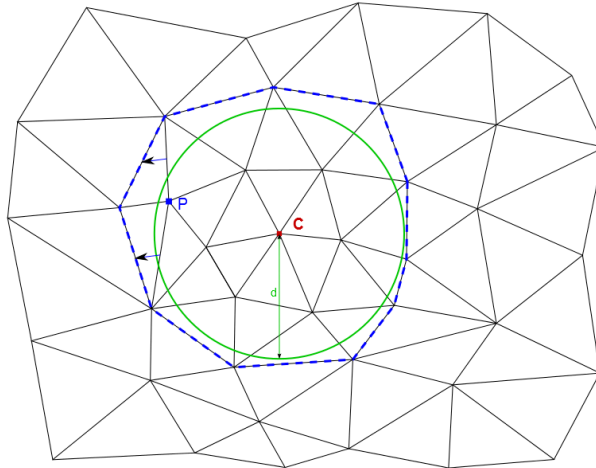


Figure 3.7. Propagation process of the first approach, starting from a center to the associated rim of an estimated crater. The contour is exclusively composed of edges of the initial triangle mesh. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

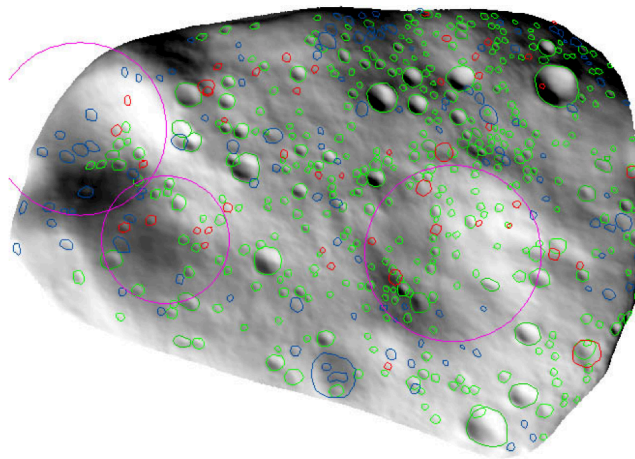


Figure 3.8. Visualization of the result of the first automatic detection method. Green circles are true positives, red ones are false positives, blue ones are missed detections and pink are primary (large) craters. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

Figure 3.9 shows a step of the second approach: after the computation of areas of interest, characterizing the regions of positive average curvature, a skeleton is extracted to keep ridge lines, by homotopic thinning (see Chapter 1, section 1.4.3). Then, circles are fitted to the vertices at the edge of the pit.

These “best fit circles” characterize the craters of the 3D model of the rock body (see Figure 3.10). After tests on the asteroid Vesta model, the authors obtain a rate of 75% detection and 10% false positive.

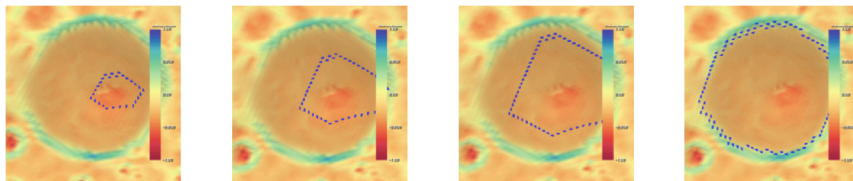


Figure 3.9. *Computation of the center of a circle and outline propagation. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip*

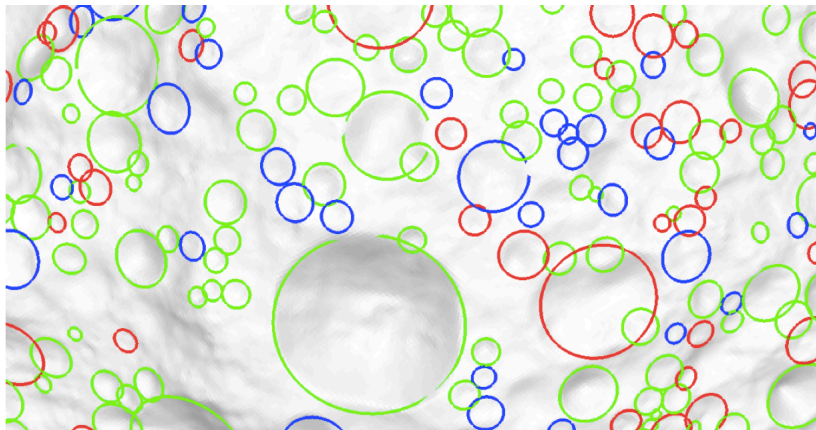


Figure 3.10. *Detection obtained from the 3D model of the asteroid Vesta. The mesh consists of more than 1,570k vertices. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip*

3.6. Botany: persistent homology to recover the branching structure of plants

In plant sciences, *phenotyping* deals with the measurement and analysis of observable global plant features, called *traits*. These traits are formed during the plant development as a result of the interaction between the genetic background of the plant and its environment (soil, weather conditions, ecosystem, etc.). Analyzing these traits enables us to understand how the plant productivity is affected by a change in either its genotype (plant breeding) or its environment, which is crucial for agriculture and food production.

In practice, the analysis of complex traits such as growth or resistance to a stress usually boils down to the measurement of well-defined parameters. Such parameters can be local, for example branch, leaf or fruit sizes and shapes. They can also be global, for example the plant architecture. We can be interested into a static analysis at a given time, for example when the plant has fully grown and given fruits, or a dynamic one, for example during the growth or during a season. The parameters that are the most difficult to quantify are often the global, dynamic ones.

In [LI 17], Mao Li *et al.* propose to use persistent homology to assess and compare branching structures of plants, enabling both to compare fully grown plants and to monitor how the branches appear along time in a given plant. Specifically, they propose to use *persistent barcodes* [GHR 08] to describe the evolving topology of a branching structure. Such a barcode records for which values of a given function connected components exist. Figure 3.11 gives an example for a height function. By convention, when two components merge, then the shortest one (with respect to the chosen function) “dies” while the longest one persists. Relevant functions to augment the branching topology with geometric feature information include not only the geodesic distance to the base of the plant, but also less intuitive ones such as the arccos (height/geodesic distance) function [LI 17].

Persistent barcodes, discretized as sets of points called *persistence diagrams*, can be compared using Wasserstein distances. Li *et al.* emphasize on the use of the *bottleneck distance* [EDE 08], which is a special case of Wasserstein distance. The bottleneck distance between two diagrams A and B

is derived from the bijections between the points of A and B . Let $d_{max}(\beta_{A,B})$ be the maximum distance between matched points in such a bijection $\beta_{A,B}$. The bottleneck distance between A and B is the shortest maximum distance $d_{max}(\beta_{A,B})$ among all bijections $\beta_{A,B}$.

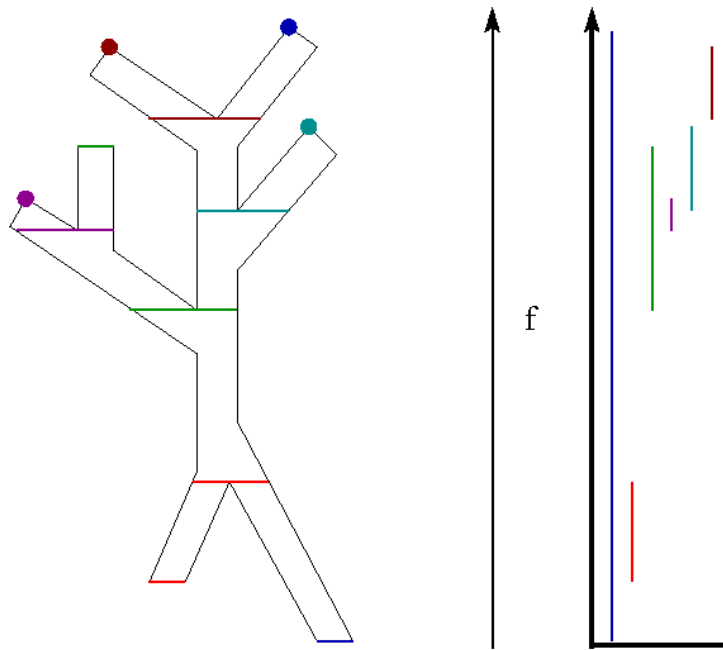


Figure 3.11. A barcode encodes as intervals, given a function f defined on a shape, the values for which each “branch” of the shape exists. For a color version of this figure, see www.iste.co.uk/mari/analysis.zip

Persistent barcodes can be used to compare plant branching structures above ground (shoots) as well as below ground (roots) [LI 17] from 3D data (voxel sets, point clouds or meshes).

Conclusion

In this book, we observed that a 3D mesh could mainly be analyzed in two ways: either by considering its geometry or by characterizing its topology. These two ways of “seeing” a 3D object or these two different angles for understanding a shape can be totally decorrelated or partially linked. Some approaches mix geometry and topology either by proposing geometric solutions that take into account some topological guarantees or by analyzing the topology and adding some geometric notions that enrich the understanding of the form.

The first half of the book was dedicated to the study of the geometry of a shape, particularly using differential geometry. The challenge lies here in the transposition of the computation of differential parameters that are not in the continuous world, but in the world of discrete surface meshes embedded in the 3D space. Furthermore, once the computation of these estimators is done, we can consider their use. Relevant geometric features can then be extracted from 3D meshes. For example, if we consider *linear* characteristics, **crest/ridge lines** can be computed [YOS 08]. They correspond to sets of vertices whose variations of curvatures are extremal, and they can be related to visual or even semantic elements. Sometimes, the context can be taken into account, and these lines can make sense in the fields of geology, planetology, anatomy, biology, etc. The algorithms for extracting these lines are numerous, and some have a consideration for the topology of the computed entities, for example, making sure to preserve the connectivity of the lines or the cycles that might exist on areas of the surface to be processed [KUD 13]. In addition to these linear attributes, it is also possible to compute *surface* characteristics on a mesh by detecting regions with similar differential properties and by

grouping the vertices according to such criteria. This is called “mesh segmentation” [SHA 08a]. Some works go further by proposing techniques to extract characteristics of structural order: graphs are used to explain the arrangement of the different similar zones, thus making it possible to highlight certain patterns in the forms [POL 17].

The second half of the book was dedicated to the study of the topology of a shape, particularly the computation of topological invariants and of the homology. Topological structures enriched with geometric notions such as Reeb graphs and Morse–Smale complexes have been presented, but it is possible to go further in the analysis of the topology with geometric considerations. For example, it is interesting to better understand the arrangement, the structure and the shape of a shape by observing the birth and death of its connected components, its holes or its cavities using a function called a “filtration”. Thus, the notion of **homological persistence** has been used in many algorithms to understand how these invariants are born, die or persist when we vary a function on the surface [EDE 17]. They assist in the detection and filtration of topological noise, the detection of structures having a geometric sense and the detection of characteristics that are difficult to observe from a purely visual point of view. Using these techniques, it is also possible to “quantify” the topology of a shape: the size of the holes, their thickness or their fragility can be calculated and thus enrich the topological analysis of geometrically relevant estimators [GON 16a].

One line of future research is to use deep-learning techniques to extract new or more robust geometric and topological mesh features.

In [GUE 18], the authors describe PCPNet, a deep-learning-based approach to estimate normal vectors and principal curvatures in a point cloud. Such local, low-level information is learnt using a patch-based architecture, adapted from the PointNet neural network [QI 17], which has been designed to operate directly on 3D point clouds. PCPNet achieves state-of-the-art results for normal vectors and curvature estimation across a wide variety of 3D point clouds. The main interest is that it does not require to tune parameters as in the algorithms described in section 1.3. Nevertheless, the authors note what they call the mismatch problem: as their network is trained only on uniform sampled point clouds, their method also does not perform in the case of changes in sampling density. Other approaches have been proposed to estimate normal vectors in a point cloud using a convolutional

neural network (CNN) [BOU 16, BEN 18]. Since such a network usually needs as input a regular grid, methods first transform the input point cloud to a new grid-based representation before applying a standard CNN framework. Although the results are usually accurate, this process may potentially result in a loss of information. To overcome this problem, many new deep-learning architectures have been recently proposed for geometric data not sampled on a regular grid (e.g. [MAR 17, MON 17, POU 18, VER 18]). Since this is still an active area of research, there is no consensus yet about which approach is the most efficient.

In [KOC 19], we can find a benchmark of seven different machine learning methods to estimate the normal vectors that are compared to five “traditional” (i.e. deterministic) ones. It is noted that deep-learning methods are superior to “traditional” ones when 3D point clouds are processed. However, if we add connectivity information, which means that 3D meshes are processed, “traditional” methods remain better. This suggests that deep-learning techniques should be based on the complete information available in a 3D mesh but very few network architectures have been proposed to represent such a structure. We can mention [Kos 18] where the network learns a local representation of the 3D mesh modeled by a first-order differential operator. It is then applied to temporal prediction of 3D mesh deformation. A research topic could be to extend this idea in order to encode the second-order differential parameters.

To the best of our knowledge, no deep-learning method has yet been proposed to recover the Betti numbers or sets of generators of a topologically complex surface. A first step towards this goal has, however, been made recently: GeoNet [HE 19] is a deep-learning architecture to recover the neighborhood information of any point in a point cloud. This paves the way towards learning global surface features directly on point cloud samplings, which we expect to be the purpose of the forthcoming research.



Bibliography

- [AGA 07] AGATHOS A., PRATIKAKIS I., PERANTONIS S. *et al.*, “3D mesh segmentation methodologies for CAD applications”, *Computer-Aided Design and Applications*, vol. 4, no. 6, pp. 827–841, CAD Solutions, 2007.
- [ALB 05] ALBOUL L., ECHEVERRIA G., “Polyhedral Gauss Maps and Curvature Characterisation of Triangle Meshes”, in MARTIN R., BEZ H., SABIN M. (eds), *Mathematics of Surfaces XI*, no. 3604 Lecture Notes in Computer Science, pp. 14–33, Springer Berlin Heidelberg, 2005.
- [ALL 03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O. *et al.*, “Anisotropic Polygonal Remeshing”, *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 485–493, ACM, July 2003.
- [ALT 13] ALTANTSETSEG E., MURAKI Y., MATSUYAMA K. *et al.*, “Feature line extraction from unorganized noisy point clouds using truncated Fourier series”, *The Visual Computer*, vol. 29, no. 6, pp. 617–626, June 2013.
- [AND 01] ANDRESEN P.R., NIELSEN M., “Non-rigid registration by geometry-constrained diffusion”, *Medical Image Analysis*, vol. 5, no. 2, pp. 81–88, 2001.
- [ANG 11] ANGELO L.D., STEFANO P.D., “Experimental comparison of methods for differential geometric properties evaluation in triangular meshes”, *Computer-Aided Design and Applications*, vol. 8, no. 2, pp. 193–210, January 2011.
- [ANO 94a] ANOSHKINA E.V., BELYAEV A.G., KUNII T.L., “Detection of ridges and ravines based on caustic singularities”, *International Journal of Shape Modeling*, vol. 1, no. 1, pp. 13–22, 1994.
- [ANO 94b] ANOSHKINA E.V., BELYAEV A.G., OKUNEV O.G. *et al.*, “Ridges and ravines: A singularity approach”, *International Journal of Shape Modeling*, vol. 1, no. 1, pp. 1–11, 1994.
- [ARM 11] ARMIN P., RONALD P., RAPHAEL F. *et al.*, “The state of the art in topology-based visualization of unsteady flow”, *Computer Graphics Forum*, vol. 30, no. 6, pp. 1789–1811, 2011.

- [ATT 06a] ATTENE M., FALCIDIENO B., SPAGNUOLO M., “Hierarchical mesh segmentation based on fitting primitives”, *The Visual Computer*, vol. 22, no. 3, pp. 181–193, Springer, 2006.
- [ATT 06b] ATTENE M., KATZ S., MORTARA M. *et al.*, “Mesh segmentation - a comparative study”, *Proceedings of the IEEE International Conference on Shape Modeling and Applications (SMI'06)*, pp. 1–7, 2006.
- [AU 08] AU O.K.-C., TAI C.-L., CHU H.-K. *et al.*, “Skeleton extraction by mesh contraction”, *ACM Transaction on Graphics*, vol. 27, no. 3, pp. 1–10, August 2008.
- [BAU 10] BAUER U., POLTHIER K., WARDETZKY M., “Uniform convergence of discrete curvatures from nets of curvature lines”, *Discrete & Computational Geometry*, vol. 43, no. 4, pp. 798–823, 2010.
- [BAY 08] BAY H., ESS A., TUYTELAARS T. *et al.*, “Speeded-up robust features (SURF)”, *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.
- [BEA 16] BEAUDET A., DUMONCEL J., DE BEER F. *et al.*, “Morphoarchitectural variation in South African fossil cercopithecoid endocasts”, *Journal of Human Evolution*, vol. 101, pp. 65–78, December 2016.
- [BEL 96] BELYAEV A.G., BOGAEVSKI I.A., KUNII T.L., Principal Direction Ridges, Technical Report no. 96-4-001, Center for Mathematical Sciences, The University of Aizu, Japan, July 1996.
- [BEL 97] BELYAEV A.G., ANOSHKINA E.V., KUNII T.L., “*Ridges, Ravines and Singularities*”, pp. 375–383, Springer Japan, Tokyo, 1997.
- [BEN 11] BENJAMIN W., POLK A.W., VISHWANATHAN S. V.N. *et al.*, “Heat walk: Robust salient segmentation of non-rigid shapes”, *Computer Graphics Forum*, vol. 30, no. 7, pp. 2097–2106, 2011.
- [BEN 18] BEN-SHABAT Y., LINDENBAUM M., FISCHER A., “Nesti-Net: Normal Estimation for Unstructured 3D Point Clouds using Convolutional Neural Networks”, 2018.
- [BER 96] BERTRAND G., “A Boolean characterization of three-dimensional simple points”, *Pattern Recognition Letters*, vol. 17, pp. 115–124, Elsevier Science Inc., 1996.
- [BER 08] BERNER A., BOKELOH M., WAND M. *et al.*, “A graph-based approach to symmetry detection”, *Proceedings of the Fifth Eurographics / IEEE VGTC Conference on Point-Based Graphics, SPBG'08*, pp. 1–8, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, 2008.
- [BER 14] BERGER M., TAGLIASACCHI A., SEVERSKY L.M. *et al.*, “State of the art in surface reconstruction from point clouds”, in LEFEBVRE S., SPAGNUOLO M. (eds), *Eurographics 2014 - State of the Art Reports*, The Eurographics Association, 2014.
- [BES 88a] BESL P.J., *Surfaces in range image understanding*, Springer Series in Perception Engineering, Springer New York, 1988.
- [BES 88b] BESL P., JAIN R., “Segmentation through variable-order surface fitting”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 2, pp. 167–192, March 1988.

- [BIA 08] BIASOTTI S., GIORGI D., SPAGNUOLO M. *et al.*, “Reeb graphs for shape analysis and applications”, *Theoretical Computer Science*, vol. 392, nos 1–3, pp. 5–22, 2008.
- [BIA 14] BIASOTTI S., FALCIDIENO B., GIORGI D. *et al.*, *Mathematical Tools for Shape Analysis and Description*, Synthesis Lectures on Computer Graphics and Animation, Morgan & Claypool Publishers, 2014.
- [BOL 11] BOLTSCHEVA D., CANINO D., MERINO-ACEITUNO S. *et al.*, “An iterative algorithm for homology computation on simplicial shapes”, *Computer-Aided Design*, vol. 43, no. 11, pp. 1457–1467, 2011.
- [BOO 88] BOOKSTEIN F.L., CUTTING C., “A proposal for the apprehension of curving craniofacial form in the three dimensions”, in MOYERS R.E., VIG K.W.L., BURDI A.R. *et al.* (eds), *Craniofacial Morphogenesis and Dysmorphogenesis*, pp. 127–140, Center for Human Growth and Development, University of Michigan, 1988.
- [BOR 84] BORGEFORS G., “Distance transformations in arbitrary dimensions”, *Computer Vision, Graphics, and Image Processing*, vol. 27, no. 3, pp. 321–345, 1984.
- [BOR 03] BORRELLI V., CAZALS F., MORVAN J.-M., “On the angular defect of triangulations and the pointwise approximation of curvatures”, *Computer Aided Geometric Design*, vol. 20, no. 6, pp. 319–341, 2003.
- [BOU 16] BOULCH A., MARLET R., “Deep learning for robust normal estimation in unstructured point clouds”, *Computer Graphics Forum*, vol. 35, no. 5, pp. 281–290, 2016.
- [BRA 85] BRADY M., PONCE J., YUILLE A. *et al.*, “Describing surfaces”, *Computer Vision, Graphics, and Image Processing*, vol. 32, no. 1, pp. 1–28, 1985.
- [BRA 10] BRAGA J., THACKERAY J.F., SUBSOL G. *et al.*, “The enamel-dentine junction in the postcanine dentition of *Australopithecus africanus*: Intra-individual metamerism and antimeric variation”, *Journal of Anatomy*, vol. 216, no. 1, pp. 62–79, January 2010.
- [BRO 12] BRONSTEIN A.M., BRONSTEIN M.M., OVSJANIKOV M., “Feature-based methods in 3D shape analysis”, in LIU Y., PEARS N., BUNTING P. (eds), *3D Imaging, Analysis, and Applications*, Chapter 5, pp. 185–219, Springer, 2012.
- [BRU 96] BRUCE J.W., GIBLIN P.J., TARI F., “Ridges, crests and sub-parabolic lines of evolving surfaces”, *International Journal of Computer Vision*, vol. 18, no. 3, pp. 195–210, June 1996.
- [BRU 99] BRUCE J.W., GIBLIN P.J., TARI F., “Families of surfaces: Focal sets, ridges and umbilics”, *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 125, no. 2, pp. 243–268, Cambridge University Press, 1999.
- [CAB 11] CABELLO S., COLIN DE VERDIÈRE É., LAZARUS F., “Finding cycles with topological properties in embedded graphs”, *SIAM Journal on Discrete Mathematics*, vol. 25, no. 4, pp. 1600–1614, 2011.
- [CAR 76] CARMO M.P.D., *Differential Geometry of Curves and Surfaces*, Prentice-Hall, 1976.
- [CAZ 05a] CAZALS F., POUGET M., “Estimating differential quantities using polynomial fitting of osculating jets”, *Computer Aided Geometric Design*, vol. 22, no. 2, pp. 121–146, 2005.

- [CAZ 05b] CAZALS F., POUGET M., “Differential topology and geometry of smooth embedded surfaces: Selected topics”, *International Journal of Computational Geometry & Applications*, vol. 15, no. 5, pp. 511–536, 2005.
- [CAZ 05c] CAZALS F., POUGET M., “Differential topology and geometry of smooth embedded surfaces: Selected topics”, *International Journal of Computational Geometry & Applications*, vol. 15, no. 5, pp. 511–536, 2005.
- [CAZ 05d] CAZALS F., POUGET M., Topology driven algorithms for ridge extraction on meshes, Research Report no. RR-5526, INRIA, 2005.
- [CAZ 06] CAZALS F., FAUGÈRE J.-C., POUGET M. *et al.*, “The implicit structure of ridges of a smooth parametric surface”, *Computer Aided Geometric Design*, vol. 23, no. 7, pp. 582–598, 2006.
- [CAZ 08a] CAZALS F., FAUGÈRE J.-C., POUGET M. *et al.*, “Ridges and umbilics of polynomial parametric surfaces”, in JÜTTLER B., PIENE R. (eds), *Geometric Modeling and Algebraic Geometry*, pp. 141–159, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [CAZ 08b] CAZALS F., POUGET M., “Algorithm 889: Jet_Fitting_3:—A generic C++ package for estimating the differential properties on sampled surfaces via polynomial fitting”, *ACM Transactions on Mathematical Software*, vol. 35, no. 3, pp. 24:1–24:20, ACM, October 2008.
- [CHE 92] CHEN X., SCHMITT F., “Intrinsic surface properties from surface triangulation”, in SANDINI G. (ed.), *Computer Vision — ECCV’92: Second European Conference on Computer Vision Santa Margherita Ligure, Italy, May 19–22, 1992 Proceedings*, Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 739–743, 1992.
- [CHE 07] CHE W., PAUL J.-C., ZHANG X., “Lines of curvature and umbilical points for implicit surfaces”, *Computer Aided Geometric Design*, vol. 24, no. 7, pp. 395–409, 2007.
- [CHE 10] CHEN C., FREEDMAN D., “Measuring and computing natural generators for homology groups”, *Computational Geometry*, vol. 43, no. 2, pp. 169–181, 2010.
- [CHE 11] CHE W., ZHANG X., ZHANG Y.-K. *et al.*, “Ridge extraction of a smooth 2-manifold surface based on vector field”, *Computer Aided Geometric Design*, vol. 28, no. 4, pp. 215–232, 2011.
- [CHE 12] CHEN X., SAPAROV A., PANG B. *et al.*, “Schelling points on 3D surface meshes”, *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 29:1–29:12, ACM, July 2012.
- [CLÉ 08] CLÉMENÇON B., BOROUCHE H., LAUG P., “Ridge extraction and its application to surface meshing”, *Engineering with Computers*, vol. 24, no. 3, pp. 287–304, Springer-Verlag, September 2008.
- [COE 14] COEURJOLLY D., LACHAUD J.-O., LEVALLOIS J., “Multigrid convergent principal curvature estimators in digital geometry”, *Computer Vision and Image Understanding*, vol. 129, pp. 27–41, 2014.
- [COH 03] COHEN-STEINER D., MORVAN J.-M., “Restricted delaunay triangulations and normal cycle”, *Proceedings of the Nineteenth Annual Symposium on Computational Geometry, SCG’03*, New York, U.S.A., ACM, pp. 312–321, 2003.

- [COL 05] COLIN DE VERDIÈRE É., LAZARUS F., “Optimal system of loops on an orientable surface”, *Discrete & Computational Geometry*, vol. 33, no. 3, pp. 507–534, 2005.
- [COL 10] COLIN DE VERDIÈRE É., ERICKSON J., “Tightening nonsimple paths and cycles on surfaces”, *SIAM Journal on Computing*, vol. 39, no. 8, pp. 3784–3813, 2010.
- [COL 17] COLIN DE VERDIÈRE É., “Computational topology of graphs on surfaces”, in GOODMAN J.E., O’ROURKE J., TÓTH C.D. (eds), *Handbook of Discrete and Computational Geometry*, Chapter 23, CRC Press LLC, 2017.
- [COR 09] CORMEN T.H., LEISERSON C.E., RIVEST R.L. *et al.*, *Introduction to Algorithms*, 3rd edition, The MIT Press, 2009.
- [CRA 13] CRANE K., DE GOES F., DESBRUN M. *et al.*, “Discrete differential geometry: An applied introduction”, *ACM SIGGRAPH 2013 Courses, SIGGRAPH’13*, New York, USA, ACM, 2013.
- [CUT 95] CUTTING C., DEAN D.P., BOOKSTEIN F.L. *et al.*, “A 3D smooth surface analysis of untreated crouzon’s syndrome in the adult”, *Journal of Craniofacial Surgery*, vol. 6, no. 6, pp. 444–453, November 1995.
- [DAM 12] DAMIAND G., DUPAS A., “Combinatorial maps for 2D and 3D image segmentation”, in BRIMKOV V.E., BARNEVA R.P. (eds), *Digital Geometry Algorithms: Theoretical Foundations and Applications to Computational Imaging*, pp. 359–393, Springer Netherlands, Dordrecht, 2012.
- [DEA 96] DEAN D., MARCUS L.F., BOOKSTEIN F.L., “Chi-square test of biological space curve affinities”, *Advances in Morphometrics*, NATO ASI Series, pp. 235–251, Springer, 1996.
- [DEA 98] DEAN D., BOOKSTEIN F.L., KONERU S. *et al.*, “Average African American three-dimensional computed tomography skull images: The potential clinical importance of ethnicity and sex”, *The Journal of Craniofacial Surgery*, vol. 9, no. 4, pp. 348–358; discussion 359, July 1998.
- [DEC 03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S. *et al.*, “Suggestive contours for conveying shape”, *Transaction on Graphics*, vol. 22, no. 3, pp. 848–855, ACM, 2003.
- [DEL 06] DELEST S., BONÉ R., CARDOT H., “Fast segmentation of triangular meshes using waterfall”, *Proceedings of the International Conference on Visualisation, Imaging and Image Processing (VIIP’06)*, pp. 308–312, 2006.
- [DEP 11] DEPARTMENT OF THE ARMY OF THE U.S.A., *Map Reading and Land Navigation: FM 3-25.26*, CreateSpace Independent Publis., 2011.
- [DEY 08] DEY T.K., LI K., SUN J. *et al.*, “Computing geometry-aware handle and tunnel loops in 3D models”, *ACM Transactions on Graphics*, vol. 27, no. 3, pp. 45:1–45:9, ACM, 2008.
- [DEY 10] DEY T.K., SUN J., WANG Y., “Approximating loops in a shortest homology basis from point data”, *ACM 26th Annual Symposium on Computational Geometry*, pp. 166–175, 2010.

- [DEY 13] DEY T.K., FAN F., WANG Y., “An efficient computation of handle and tunnel loops via Reeb graphs”, *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 32:1–32:10, ACM, July 2013.
- [DON 05] DONG C.-S., WANG G.-Z., “Curvatures estimation on triangular mesh”, *Journal of Zhejiang University-SCIENCE A*, vol. 6, no. 1, pp. 128–136, 2005.
- [DUM 03] DUMAS J.-G., HECKENBACH F., SAUNDERS D. *et al.*, “Computing simplicial homology based on efficient smith normal form algorithms”, pp. 177–206, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.
- [DYN 01] DYN N., HORMANN K., KIM S.-J. *et al.*, “Optimizing 3D triangulations using discrete curvature analysis”, in LYCHE T., SCHUMAKER L.L. (eds), *Mathematical Methods for Curves and Surfaces*, pp. 135–146, Vanderbilt University, Nashville, TN, USA, 2001.
- [EBE 94] EBERLY D., GARDNER R., MORSE B. *et al.*, “Ridges for image analysis”, *Journal of Mathematical Imaging and Vision*, vol. 4, no. 4, pp. 353–373, December 1994.
- [EDE 06] EDELSBRUNNER H., ABLOWITZ M.J., DAVIS S.H. *et al.*, *Geometry and Topology for Mesh Generation (Cambridge Monographs on Applied and Computational Mathematics)*, Cambridge University Press, New York, USA, 2006.
- [EDE 08] EDELSBRUNNER H., HARER J., “Persistent Homology – a Survey”, in J.E. GOODMAN J.P., POLLACK R. (eds), *Surveys on Discrete and Computational Geometry: Twenty Years Later*, vol. 453, Providence, Rhode Island, Amer. Math. Soc., Contemporary Mathematics, pp. 257–282, 2008.
- [EDE 17] EDELSBRUNNER H., MOROZOV D., “Persistent homology”, in GOODMAN J.E., O’ROURKE J., TÓTH C.D. (eds), *Handbook of Discrete and Computational Geometry*, Chapter 24, CRC Press LLC, 2017.
- [EPS 11] EPSTEIN C., CARLSSON G., EDELSBRUNNER H., “Topological data analysis”, *Inverse Problems*, vol. 27, no. 12, p. 120201, 2011.
- [ERI 05] ERICKSON J., WHITTLESEY K., “Greedy optimal homotopy and homology generators”, *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005*, pp. 1038–1046, Vancouver, British Columbia, Canada, January 23–25, 2005.
- [FAR 98] FAROUKI R.T., “On integrating lines of curvature”, *Computer Aided Geometric Design*, vol. 15, no. 2, pp. 187–192, 1998.
- [FAS 16] FASSETT C.I., “Analysis of impact crater populations and the geochronology of planetary surfaces in the inner solar system”, *Journal of Geophysical Research Planets*, vol. 121, pp. 1900–1926, 2016.
- [FEN 13] FENG X., TONG Y., “Choking loops on surfaces”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 8, pp. 1298–1306, IEEE Educational Activities Department, 2013.
- [FER 16] FERRI M., “Progress in persistence for shape analysis (extended abstract)”, in BAC A., MARI J.-L. (eds), *Computational Topology in Image Context: 6th International Workshop, CTIC 2016, Marseille, France, June 15–17, 2016, Proceedings*, pp. 3–6, Springer International Publishing, Marseille, 2016.

- [FID 97] FIDRICH M., “Following feature lines across scale”, in TER HAAR ROMENY B., FLORACK L., KOENDERINK J. *et al.* (eds), *Scale-Space Theory in Computer Vision*, Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 140–151, 1997.
- [FLO 15] FLORIANI L.D., FUGACCI U., IURICICH F. *et al.*, “Morse complexes for shape segmentation and homological analysis: Discrete models and algorithms”, *Computer Graphics Forum*, vol. 34, no. 2, pp. 761–785, 2015.
- [FOR 98] FORMAN R., “Morse theory for cell complexes”, *Advances in Mathematics*, vol. 134, no. 1, pp. 90–145, 1998.
- [FRO 90] FROSINI P., “A distance for similarity classes of submanifolds of a Euclidean space”, *Bulletin of the Australian Mathematical Society*, vol. 42, pp. 407–415, 12 1990.
- [GAL 06] GAL R., COHEN-OR D., “Salient geometric features for partial shape matching and similarity”, *ACM Transactions on Graphics*, vol. 25, no. 1, pp. 130–150, ACM, January 2006.
- [GAL 09] GALL J., STOLL C., DE AGUIAR E. *et al.*, “Motion capture using joint skeleton tracking and surface estimation”, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’09)*, IEEE Computer Society, pp. 1746–1753, June 2009.
- [GAT 06] GATZKE T.D., GRIMM C.M., “Estimating curvature on triangular meshes”, *International Journal of Shape Modeling*, vol. 12, no. 1, pp. 1–28, 2006.
- [GHR 08] GHRIST R., “Barcodes: The persistent topology of data”, *Bulletin of the American Mathematical Society*, vol. 45, pp. 61–75, 2008.
- [GIR 00] GIRSHICK A., INTERRANTE V., HAKER S. *et al.*, “Line direction matters: An argument for the use of principal directions in 3D line drawings”, *Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering, NPAR’00*, New York, USA, ACM, pp. 43–52, 2000.
- [GOL 05] GOLDMAN R., “Curvature formulas for implicit curves and surfaces”, *Computer Aided Geometric Design*, vol. 22, no. 7, pp. 632–658, 2005.
- [GOL 08] GOLOVINSKIY A., FUNKHOUSER T., “Randomized cuts for 3D mesh analysis”, *ACM Transactions on Graphics (Proceeding SIGGRAPH Asia)*, vol. 27, no. 5, December 2008.
- [GON 16a] GONZALEZ-LORENZO A., BAC A., MARI J. *et al.*, “Two measures for the homology groups of binary volumes”, *Discrete Geometry for Computer Imagery - 19th IAPR International Conference, DGCI 2016, Nantes, France, April 18–20, 2016. Proceedings*, pp. 154–165, 2016.
- [GON 16b] GONZALEZ-LORENZO A., JUDA M., BAC A. *et al.*, “Fast, simple and separable computation of betti numbers on three-dimensional cubical complexes”, *Computational Topology in Image Context - 6th International Workshop, CTIC 2016, Marseille, France, June 15–17, 2016, Proceedings*, pp. 130–139, 2016.
- [GON 17] GONZALEZ-LORENZO A., BAC A., MARI J.-L. *et al.*, “Allowing cycles in discrete Morse theory”, *Topology and its Applications*, vol. 228, Elsevier, September 2017.
- [GUÉ 93] GUÉZIEC A., “Large deformable splines, crest lines and matching”, *1993 (4th) International Conference on Computer Vision*, pp. 650–657, May 1993.

- [GUÉ 94] GUÉZIEC A., AYACHE N., “Smoothing and matching of 3D space curves”, *International Journal of Computer Vision*, vol. 12, no. 1, pp. 79–104, February 1994.
- [GUÉ 95] GUÉZIEC A., “Surface representation with deformable splines: Using decoupled variables”, *IEEE Computational Science and Engineering*, vol. 2, no. 1, pp. 69–80, Spring 1995.
- [GUE 18] GUERRERO P., KLEIMAN Y., OVSJANIKOV M. *et al.*, “PCPNet learning local shape properties from raw point clouds”, *Computer Graphics Forum*, vol. 37, no. 2, pp. 75–85, 2018.
- [GUM 01] GUMHOLD S., WANG X., MACLEOD R., “Feature extraction from point clouds”, *In Proceedings of the 10th International Meshing Roundtable*, pp. 293–305, October 2001.
- [GUO 14] GUO Y., BENNAMOUN M., SOHEL F.A. *et al.*, “3D object recognition in cluttered scenes with local surface features: A survey”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2270–2287, 2014.
- [GYU 06] GYULASSY A., NATARAJAN V., PASCUCCI V. *et al.*, “A topological approach to simplification of 3D scalar functions”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 4, pp. 474–484, 2006.
- [HAG 92] HAGEN H., HAHMANN S., “Generalized focal surfaces: A new method for surface interrogation”, *Proceedings Visualization’92*, pp. 70–76, October 1992.
- [HAH 08] HAHMANN S., BELYAEV A., BUSE L. *et al.*, “Shape interrogation”, in DE FLORIANI L., SPAGNUOLO M. (eds), *Shape Analysis and Structuring, Mathematics and Visualization*, Chapter 1, pp. 1–52, Springer, Berlin, Germany, 2008.
- [HAM 93] HAMANN B., “*Geometric Modelling*”, Chapter Curvature Approximation for Triangulated Surfaces, pp. 139–153, Springer Vienna, Vienna, 1993.
- [HAR 83] HARALICK R.M., WATSON L.T., LAFFEY T.J., “The topographic primal sketch”, *The International Journal of Robotics Research*, vol. 2, no. 1, pp. 50–72, 1983.
- [HAR 88] HARRIS C., STEPHENS M., “A combined corner and edge detector”, *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, 1988.
- [HAT 02] HATCHER A., *Algebraic Topology*, Cambridge University Press, Cambridge, New York, 2002.
- [HE 19] HE T., HUANG H., YI L. *et al.*, “GeoNet: Deep geodesic networks for point cloud analysis”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [HÉT 03] HÉTOY F., ATTALI D., “Detection of constrictions on closed polyhedral surfaces”, *VisSym 2003, Symposium on Visualization*, pp. 67–74, Grenoble, France, May 26–28, 2003.
- [HIL 05] HILDEBRANDT K., POLTHIER K., WARDETZKY M., “Smooth feature lines on surface meshes”, *Proceedings of the Third Eurographics Symposium on Geometry Processing, SGP’05*, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, 2005.
- [HOS 92] HOSAKA M., *Modeling of Curves and Surfaces in CAD/CAM*, Computer Graphics: Systems and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

- [HUB 01] HUBELI A., GROSS M., “Multiresolution feature extraction for unstructured meshes”, *Visualization, 2001. VIS '01. Proceedings*, pp. 287–294, 2001.
- [HUI 07] HUI A., DE FLORIANI L., “A two-level topological decomposition for non-manifold simplicial shapes”, *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling (SPM)*, pp. 355–360, 2007.
- [IAR 15] IARUSSI E., BOMMES D., BOUSSEAU A., “BendFields: Regularized curvature fields from rough concept sketches”, *ACM Transactions on Graphics*, vol. 34, no. 3, pp. 24:1–24:16, ACM, May 2015.
- [ITT 98] ITTI L., KOCH C., NIEBUR E., “A model of saliency-based visual attention for rapid scene analysis”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, IEEE Computer Society, 1998.
- [JIA 08] JIAO X., BAYYANA N.R., “Identification of C^1 and C^2 discontinuities for surface meshes in CAD”, *Computer-Aided Design*, vol. 40, no. 2, pp. 160–175, 2008.
- [JIN 05] JIN S., LEWIS R.R., WEST D., “A comparison of algorithms for vertex normal computation”, *The Visual Computer*, vol. 21, no. 1, pp. 71–82, 2005.
- [JIN 06] JING H., ZHANG W., ZHOU B., “Ridge-valley lines smoothing and optimizing”, in PAN Z., CHEOK A., HALLER M. *et al.* (eds), *Advances in Artificial Reality and Tele-Existence*, Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 502–511, 2006.
- [JOH 99] JOHNSON A., HEBERT M., “Using spin images for efficient object recognition in cluttered 3D scenes”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.
- [JOO 14] JOO H.K., YAZAKI T., TAKEZAWA M. *et al.*, “Differential geometry properties of lines of curvature of parametric surfaces and their visualization”, *Graphical Models*, vol. 76, no. 4, pp. 224–238, 2014.
- [KAC 04] KACZYNSKI T., MISCHAIKOW K., MROZEK M., *Computational Homology*, vol. 157 of *Applied Mathematical Sciences*, Springer-Verlag, New York, 2004.
- [KAL 96] KALVIN A.D., TAYLOR R.H., “Superfaces: Polygonal mesh simplification with bounded error”, *Computer Graphics and Applications*, vol. 16, no. 3, pp. 64–77, IEEE, 1996.
- [KÄL 07a] KÄLBERER F., NIESER M., POLTHIER K., “QuadCover - surface parameterization using branched coverings”, *Computer Graphics Forum*, vol. 26, no. 3, pp. 375–384, Blackwell Publishing Ltd, 2007.
- [KAL 07b] KALOGERAKIS E., SIMARI P., NOWROUZEZHRAI D. *et al.*, “Robust statistical estimation of curvature on discretized surfaces”, *Proceedings of the Fifth Eurographics Symposium on Geometry Processing, SGP'07*, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, pp. 13–22, 2007.
- [KAL 09] KALOGERAKIS E., NOWROUZEZHRAI D., SIMARI P. *et al.*, “Extracting lines of curvature from noisy point clouds”, *Computer-Aided Design*, vol. 41, no. 4, pp. 282–292, 2009.

- [KAL 10] KALOGERAKIS E., HERTZMANN A., SINGH K., “Learning 3D mesh segmentation and labeling”, *ACM Transactions on Graphics*, vol. 29, no. 4, pp. 102:1–102:12, ACM, July 2010.
- [KAT 03] KATZ S., TAL A., “Hierarchical mesh decomposition using fuzzy clustering and cuts”, *Transaction on Graphics*, vol. 22, no. 3, pp. 954–961, ACM, 2003.
- [KAU 87] KAUFMAN A., “Efficient algorithms for 3D scan-conversion of parametric curves, surfaces, and volumes”, *SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 171–179, ACM, August 1987.
- [KAZ 03] KAZHDAN M., FUNKHOUSER T., RUSINKIEWICZ S., “Rotation invariant spherical harmonic representation of 3D shape descriptors”, *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, SGP’03*, Eurographics Association, pp. 156–164, 2003.
- [KEN 96] KENT J.T., MARDIA K.V., WEST J.M., “Ridge curves and shape analysis”, *Proceedings of the British Machine Vision Conference 1996, BMVC 1996*, University of Edinburgh, UK, 1996, pp. 1–10, 1996.
- [KHA 98] KHANEJA N., MILLER M., GRENANDER U., “Dynamic programming generation of curves on brain surfaces”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1260–1265, November 1998.
- [KIM 05] KIM S.-K., KIM C.-H., “Finding ridges and valleys in a discrete surface using a modified MLS approximation”, *Computer-Aided Design*, vol. 37, no. 14, pp. 1533–1542, 2005.
- [KIM 06] KIM S.-K., KIM C.-H., “Finding ridges and valleys in a discrete surface using a modified MLS approximation”, *Computer-Aided Design*, vol. 38, no. 2, pp. 173–180, 2006.
- [KNO 10] KNOPP J., PRASAD M., WILLEMS G. *et al.*, “Hough transform and 3D SURF for robust three dimensional classification”, *Proceedings of the 11th European Conference on Computer Vision*, pp. 589–602, 2010.
- [KOC 19] KOCH S., MATVEEV A., JIANG Z. *et al.*, “ABC: A big CAD model dataset for geometric deep learning”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [KOE 92] KOENDERINK J.J., VAN DOORN A.J., “Surface shape and curvature scales”, *Image and Vision Computing*, vol. 10, no. 8, pp. 557–564, 1992.
- [KOE 93] KOENDERINK J.J., VAN DOORN A.J., “Local features of smooth shapes: Ridges and courses”, *Proceedings SPIE*, vol. 2031, pp. 2031-2031-12, 1993.
- [Kos 18] KOSTRIKOV I., JIANG Z., PANOZZO D. *et al.*, “Surface networks”, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2540–2548, June 2018.
- [KUD 11] KUDELSKI D., VISEUR S., SCROFANI G. *et al.*, “Feature line extraction on meshes through vertex marking and 2D topological operators”, *International Journal of Image and Graphics*, vol. 11, no. 4, pp. 531–548, 2011.
- [KUD 13] KUDELSKI D., VISEUR S., MARI J., “Skeleton extraction of vertex sets lying on arbitrary triangulated 3D meshes”, *Discrete Geometry for Computer Imagery - 17th IAPR International Conference, DGCI 2013*, pp. 203–214, Seville, Spain, March 20–22, 2013.

- [KWE 94] KWEON I., KANADE T., “Extracting topographic terrain features from elevation maps”, *CVGIP: Image Understanding*, vol. 59, no. 2, pp. 171–182, 1994.
- [LAU 16] LAU M., DEV K., SHI W. *et al.*, “Tactile mesh saliency”, *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 52:1–52:11, ACM, July 2016.
- [LAV 05] LAVOUÉ G., DUPONT F., BASKURT A., “A new CAD mesh segmentation method based on curvature tensor analysis”, *Computer-Aided Design*, vol. 37, no. 10, pp. 975–987, 2005.
- [LAV 09a] LAVOUÉ G., “A local roughness measure for 3D meshes and its application to visual masking”, *ACM Transactions on Applied Perception*, vol. 5, no. 4, pp. 21:1–21:23, ACM, 2009.
- [LAV 09b] LAVOUÉ G., “A local roughness measure for 3D meshes and its application to visual masking”, *ACM Transactions on Applied Perception*, vol. 5, no. 4, pp. 21:1–21:23, ACM, February 2009.
- [LEE 94] LEE T., KASHYAP R., CHU C., “Building skeleton models via 3D medial surface/axis thinning algorithms”, *Graphical Models and Image Processing*, vol. 56, no. 6, pp. 462–478, November 1994.
- [LEE 05] LEE C.H., VARSHNEY A., JACOBS D.W., “Mesh saliency”, *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 659–666, ACM, 2005.
- [LEI 16] LEIFMAN G., SHTROM E., TAL A., “Surface regions of interest for viewpoint selection”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 12, pp. 2544–2556, 2016.
- [LÉO 09] LÉON J.-C., DE FLORIANI L., HÉTROUY F., “Classification of non-manifold singularities from transformations of 2-manifolds”, *SMI 2009 - IEEE International Conference on Shape Modeling and Applications*, Beijing, China, IEEE, pp. 179–184, June 2009.
- [LÉV 10] LÉVY B., ZHANG H.R., “Spectral mesh processing”, *ACM SIGGRAPH 2010 Courses, SIGGRAPH’10*, pp. 8:1–8:312, New York, USA, ACM, 2010.
- [LEV 12] LEVINE J.A., PAULSEN R.R., ZHANG Y., “Mesh processing in medical-image analysis - a tutorial”, *IEEE Computer Graphics and Applications*, vol. 32, no. 5, pp. 22–28, September 2012.
- [LI 11] LI E., LÉVY B., ZHANG X. *et al.*, “Meshless quadrangulation by global parameterization”, *Computers & Graphics*, vol. 35, no. 5, pp. 992–1000, 2011.
- [LI 17] LI M., DUNCAN K., TOPP C.N. *et al.* “Persistent homology and the branching topologies of plants”, *American Journal of Botany*, vol. 104, no. 3, pp. 349–353, 2017.
- [LIU 04] LIU R., ZHANG H., “Segmentation of 3D meshes through spectral clustering”, *Proceedings of the Pacific Conference on Computer Graphics and Applications (PG’04)*, pp. 298–305, 2004.
- [LIU 07] LIU D., XU G., “Angle deficit approximation of Gaussian curvature and its convergence over quadrilateral meshes”, *Computer-Aided Design*, vol. 39, no. 6, pp. 506–517, 2007.

- [LO 09] LO T.-W.R., SIEBERT J.P., “Local feature extraction and matching on range images: 2.5D SIFT”, *Computer Vision and Image Understanding*, vol. 113, no. 12, pp. 1235–1250, 2009.
- [LOW 04] LOWE D., “Distinctive image features from scale-invariant keypoints”, *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [LUK 98] LUKÁCS G., ANDOR L., “Computing natural division lines on free-form surfaces based on measured data”, *Proceedings of the International Conference on Mathematical Methods for Curves and Surfaces II Lillehammer, 1997*, Nashville, TN, USA, Vanderbilt University, pp. 319–326, 1998.
- [MAE 96] MAEKAWA T., WOLTER F.-E., PATRIKALAKIS N.M., “Umbilics and lines of curvature for shape interrogation”, *Computer Aided Geometric Design*, vol. 13, no. 2, pp. 133–161, 1996.
- [MAG 07] MAGID E., SOLDEA O., RIVLIN E., “A comparison of Gaussian and mean curvature estimation methods on triangular meshes of range image data”, *Computer Vision and Image Understanding*, vol. 107, no. 3, pp. 139–159, 2007.
- [MAR 80] MARR D., HILDRETH E.C., “Theory of edge detection”, *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 207, no. 1167, pp. 187–217, 1980.
- [MAR 83] MARTIN R., “Principal patches - a new class of surface patch based on differential geometry”, in TEN HAGEN P. (ed.), *Eurographics Conference Proceedings*, The Eurographics Association, 1983.
- [MAR 09] MARI J.-L., “Surface sketching with a voxel-based skeleton”, *15th IAPR International Conference on Discrete Geometry for Computer Imagery (DGCI'09)*, vol. 5810 of *Lecture Notes in Computer Science*, pp. 325–336, Springer, 2009.
- [MAR 10] MARCHI S., BARBIERI C., KUEPPERS M. *et al.*, “The cratering history of asteroid (2867) Steins”, *Planetary and Space Science*, vol. 58, no. 1116, pp. 65–78, 2010.
- [MAR 17] MARON H., GALUN M., AIGERMAN N. *et al.*, “Convolutional neural networks on surfaces via seamless toric covers”, *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.
- [MAR 19] MARI J.-L., VISEUR S., BOULEY S. *et al.*, *Robust Detection of Circular Shapes on 3D Meshes Based on Discrete Curvatures - Application to Impact Craters Recognition*, AGU Geophysical Monograph Series, American Geophysical Union, 2019.
- [MAX 99] MAX N., “Weights for computing vertex normals from facet normals”, *Journal of Graphics Tools*, vol. 4, no. 2, pp. 1–6, 1999.
- [MEE 00] MEEK D.S., WALTON D.J., “On surface normal and Gaussian curvature approximations given data sampled from a smooth surface”, *Computer Aided Geometric Design*, vol. 17, no. 6, pp. 521–543, July 2000.
- [MÉR 11] MÉRIGOT Q., OVSJANIKOV M., GUIBAS L.J., “Voronoi-based curvature and feature estimation from point clouds”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 6, pp. 743–756, June 2011.
- [MES 07] MESMOUDI M.M., DANOVARO E., FLORIANI L.D. *et al.*, “Surface segmentation through concentrated curvature”, *14th International Conference on Image Analysis and Processing (ICIAP 2007)*, pp. 671–676, September 2007.

- [MES 12] MESMOUDI M.M., FLORIANI L.D., MAGILLO P., “Discrete curvature estimation methods for triangulated surfaces”, *Applications of Discrete Geometry and Mathematical Morphology*, pp. 28–42, Springer, Berlin, Heidelberg, 2012.
- [MES 18] MESNIL R., DOUTHE C., BAVEREL O. *et al.*, “Morphogenesis of surfaces with planar lines of curvature and application to architectural design”, *Automation in Construction*, vol. 95, pp. 129–141, 2018.
- [MEY 03] MEYER M., DESBRUN M., SCHRÖDER P. *et al.*, “Discrete differential-geometry operators for triangulated 2-manifolds”, in HEGE H.-C., POLTHIER K. (eds), *Visualization and Mathematics III*, Mathematics and Visualization, pp. 35–57, Springer Berlin Heidelberg, 2003.
- [MIS 13] MISCHAIKOW K., NANDA V., “Morse theory for filtrations and efficient computation of persistent homology”, *Discrete & Computational Geometry*, vol. 50, no. 2, pp. 330–353, Springer US, 2013.
- [MIT 04] MITRA N.J., NGUYEN A., GUIBAS L., “Estimating surface normals in noisy point cloud data”, *International Journal of Computational Geometry Estimating Surface Normals in Noisy Point Cloud Data and Applications*, vol. 14, nos 4–5, pp. 261–276, 2004.
- [MOK 01] MOKHTARIAN F., KHALILI N., YUEN P., “Curvature computation on free-form 3D meshes at multiple scales”, *Computer Vision and Image Understanding*, vol. 83, no. 2, pp. 118–139, 2001.
- [MON 68] MONTANARI U., “A Method for obtaining skeletons using a quasi-euclidean distance”, *Journal of the ACM*, vol. 15, no. 4, pp. 600–624, 1968.
- [MON 92] MONGA O., BENAYOUN S., FAUGERAS O.D., “From partial derivatives of 3D density images to ridge lines”, *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 354–359, June 1992.
- [MON 95] MONGA O., BENAYOUN S., “Using partial derivatives of 3D images to extract typical surface features”, *Computer Vision and Image Understanding*, vol. 61, no. 2, pp. 171–189, 1995.
- [MON 17] MONTI F., BOSCAINI D., MASCI J. *et al.*, “Geometric deep learning on graphs and manifolds using mixture model CNNs”, *CVPR - IEEE Conference on Computer Vision & Pattern Recognition*, 2017.
- [MOR 96] MORRIS R., “The sub-parabolic lines of a surface”, *Proceedings of the 6th IMA Conference on the Mathematics of Surfaces*, pp. 79–102, New York, USA, Clarendon Press, 1996.
- [MOR 10] MORITA I., SAKAMOTO H., “Anisotropic remeshing based on topology of curvature-lines and isoparametric-lines of 3D surface”, *International Workshop on Advanced Image Technology, IWAIT 2010*, January 2010.
- [MUK 00] MUKHERJEE J., DAS P.P., KUMAR M.A. *et al.*, “On approximating Euclidean metrics by digital distances in 2D and 3D”, *Pattern Recognition Letters*, vol. 21, nos 6–7, pp. 573–582, 2000.
- [MUN 84] MUNKRES J.R., *Elements of Algebraic Topology*, Addison-Wesley, 1984.

- [MUS 11] MUSUVATHY S., COHEN E., DAMON J. *et al.*, “Principal curvature ridges and geometrically salient regions of parametric B-spline surfaces”, *Computer-Aided Design*, vol. 43, no. 7, pp. 756–770, Butterworth-Heinemann, July 2011.
- [NAC 85] NACKMAN L.R., PIZER S.M., “Three-Dimensional Shape Description Using the Symmetric Axis Transform I: Theory”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, no. 2, pp. 187–202, March 1985.
- [NAD 16a] NADER G., WANG K., HÉTROUY-WHEELER F. *et al.*, “Just noticeable distortion profile for flat-shaded 3D mesh surfaces”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 11, pp. 2423–2436, 2016.
- [NAD 16b] NADER G., WANG K., HÉTROUY-WHEELER F. *et al.*, “Visual contrast sensitivity and discrimination for 3D meshes and their applications”, *Computer Graphics Forum*, vol. 35, no. 7, pp. 497–506, 2016.
- [NAT 06] NATARAJAN V., WANG Y., BREMER P. *et al.*, “Segmenting molecular surfaces”, *Computer Aided Geometric Design*, vol. 23, no. 6, pp. 495–509, 2006.
- [NIK 12] NIKOLIĆ M., “Measuring similarity of graph nodes by neighbor matching”, *Intelligent Data Analysis*, vol. 16, no. 6, pp. 865–878, 2012.
- [NOR 09] NORMAND N., ÉVENOU P., “Medial axis lookup table and test neighborhood computation for 3D chamfer norms”, *Pattern Recognition*, vol. 42, no. 10, pp. 2288–2296, 2009.
- [OHT 04] OHTAKE Y., BELYAEV A., SEIDEL H.-P., “Ridge-valley lines on meshes via implicit surface fitting”, *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 609–612, ACM, August 2004.
- [OXF 17] OXFORD DICTIONARIES, “Definition of feature in English”. Available at: <https://en.oxforddictionaries.com/definition/feature>, 2017. [Online; accessed 16-August-2017].
- [PAG 02] PAGE D.L., SUN Y., KOSCHAN A.F. *et al.*, “Normal vector voting: crease detection and curvature estimation on large, noisy meshes”, *Graphical Models*, vol. 64, no. 3, pp. 199–229, May 2002.
- [PAG 06] PAGE D.L., KOSCHAN A.F., ABIDI M.A., “Linking feature lines on 3D triangle meshes with artificial potential fields”, *3D Data Processing, Visualization, and Transmission, Third International Symposium on*, pp. 358–364, June 2006.
- [PAN 10a] PANG X., SONG Z., PANG M., “Extraction of the lines of curvature from raw point cloud”, *Proceedings of the 9th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry, VRCAI'10*, pp. 225–228, New York, USA, ACM, December 2010.
- [PAN 10b] PANOZZO D., PUPPO E., ROCCA L., “Efficient multi-scale curvature and crease estimation”, *2nd International Workshop on Computer Graphics, Computer Vision and Mathematics, GraVisMa 2010 - Workshop Proceedings*, pp. 9–16, 2010.
- [PAR 12] PARK M.K., LEE S.J., LEE K.H., “Multi-scale tensor voting for feature extraction from unstructured point clouds”, *Graphical Models*, vol. 74, no. 4, pp. 197–208, 2012.

- [PAT 10] PATRIKALAKIS N.M., MAEKAWA T., *Shape Interrogation for computer aided design and manufacturing*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, Hyperbook edition: <http://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/mathe.html>.
- [PAU 03] PAULY M., KEISER R., GROSS M., “Multi-scale feature extraction on point-sampled surfaces”, *Computer Graphics Forum*, vol. 22, no. 3, pp. 281–289, 2003.
- [PER 02] PERELMAN G., “The entropy formula for the Ricci flow and its geometric applications”, 2002.
- [PER 03a] PERELMAN G., “Finite extinction time for the solutions to the Ricci flow on certain three-manifolds”, 2003.
- [PER 03b] PERELMAN G., “Ricci flow with surgery on three-manifolds”, 2003.
- [PET 02] PETITJEAN S., “A survey of methods for recovering quadrics in triangle meshes”, *ACM Computing Surveys*, vol. 34, no. 2, pp. 211–262, ACM, June 2002.
- [POI 95] POINCARÉ H., “Analysis situs”, *Journal de l’Ecole Polytechnique*, vol. 2, no. 1, pp. 1–123, 1895. Available at: <http://gallica.bnf.fr/ark:/12148/bpt6k4337198/f7.image>.
- [POL 15a] POLETTE A., Analyse de maillages surfaciques par construction et comparaison de modèles moyens et par décomposition par graphes s’appuyant sur les courbures discrètes : Application à l’étude de la cornée humaine, PhD thesis, Aix-Marseille Université, Marseille, France, 2015.
- [POL 15b] POLETTE A., MEUNIER J., MARI J.-L., “Feature extraction using a shape descriptor graph based on discrete curvature patches”, *Computer Graphics International, CGI 2015*, Strasbourg, France, 2015.
- [POL 17] POLETTE A., MEUNIER J., MARI J., ““Shape-Curvature-Graph”: Towards a new model of representation for the description of 3D meshes”, *Augmented Reality, Virtual Reality, and Computer Graphics - 4th International Conference, AVR 2017, Ugento, Italy, June 12-15, 2017, Proceedings, Part II*, vol. 10325 of *Lecture Notes in Computer Science*, Springer, pp. 369–384, 2017.
- [POR 71] PORTEOUS I.R., “The normal singularities of a submanifold”, *Journal of Differential Geometry*, vol. 5, no. 3–4, pp. 543–564, Lehigh University, 1971.
- [POR 01] PORTEOUS I.R., *Geometric Differentiation: For the Intelligence of Curves and Surfaces*, Cambridge University Press, Cambridge, UK, New York, 2nd edition, December 2001.
- [POT 07] POTTMANN H., WALLNER J., YANG Y.-L. *et al.*, “Principal curvatures from the integral invariant viewpoint”, *Computer Aided Geometric Design*, vol. 24, no. 8, pp. 428–442, 2007.
- [POT 09] POTTMANN H., WALLNER J., HUANG Q.-X. *et al.*, “Integral invariants for robust geometry processing”, *Computer Aided Geometric Design*, vol. 26, no. 1, pp. 37–60, 2009.
- [POU 18] POULENARD A., OVSJANIKOV M., “Multi-directional geodesic neural networks via equivariant convolution”, *ACM Transactions on Graphics (SIGGRAPH Asia 2018 Proceedings)*, vol. 37, 2018.
- [PRE 07] PRESS W.H., TEUKOLSKY S.A., VETTERLING W.T. *et al.*, “Numerical recipes - the art of scientific computing”, September 2007. Available at: <http://numerical.recipes/>.

- [QI 17] QI C.R., SU H., MO K. *et al.*, “PointNet: Deep learning on point sets for 3D classification and segmentation”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [REE 46] REEB G., “Sur les points singuliers d’une forme de Pfaff complètement intégrable ou d’une fonction numérique”, *Comptes rendus de l’Académie des Sciences*, vol. 222, pp. 847–849, 1946.
- [REN 12] JANSEN VAN RENSBURG G.J., WILKE D.N., KOK S., “Human skull shape and masticatory induced stress: Objective comparison through the use of non-rigid registration”, *International Journal for Numerical Methods in Biomedical Engineering*, vol. 28, no. 1, pp. 170–185, 2012.
- [ROD 18] RODRIGUES R. S.V., MORGADO J. F.M., GOMES A. J.P., “Part-based mesh segmentation: A survey”, *Computer Graphics Forum*, vol. 37, no. 6, pp. 235–274, 2018.
- [ROS 75] ROSENFELD A., “A characterization of parallel thinning algorithms”, *Information Control*, vol. 29, pp. 286–291, 1975.
- [ROS 82] ROSENFELD A., KAK A., *Digital Pictures Processing*, 2nd edition, Academic Press, 1982.
- [RÖS 00a] RÖSSL C., KOBBELT L., “Line-art rendering of 3D-models”, *Proceedings of the 8th Pacific Conference on Computer Graphics and Applications, PG’00*, p. 87, Washington, DC, USA, IEEE Computer Society, 2000.
- [RÖS 00b] RÖSSL C., KOBBELT L., SEIDEL H.-P., “Extraction of feature lines on triangulated surfaces using morphological operators”, *AAAI Spring Symposium on Smart Graphics*, vol. 4, pp. 71–75, March 2000.
- [ROU 87] ROUSSEEUW P.J., LEROY A.M., *Robust Regression and Outlier Detection*, John Wiley & Sons, Inc., New York, USA, 1987.
- [RUS 04] RUSINKIEWICZ S., “Estimating curvatures and their derivatives on triangle meshes”, *Proceedings 2nd International Symposium on 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004*, pp. 486–493, September 2004.
- [SAN 90] SANDER P.T., ZUCKER S.W., “Inferring surface trace and differential structure from 3D images”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 9, pp. 833–854, September 1990.
- [SAN 92] SANDER P.T., ZUCJER S.W., “Singularities of principal direction fields from 3D images”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 3, pp. 309–317, March 1992.
- [SAN 09] SANSONI G., TREBESCHI M., DOCCHIO F., “State-of-the-art and applications of 3D imaging sensors in industry, cultural heritage, medicine, and criminal investigation”, *Sensors*, vol. 9, no. 1, pp. 568–601, 2009.
- [SEE 16] SEEMANN P., FUHRMANN S., GUTHE S. *et al.*, “Simplification of multi-scale geometry using adaptive curvature fields”, *CoRR*, vol. abs/1610.07368, 2016.
- [SER 99] SERGERAERT F., “Constructive algebraic topology”, *ACM SIGSAM Bulletin*, vol. 33, pp. 13–25, 1999.

- [SHA 08a] SHAMIR A., “A survey on mesh segmentation techniques”, *Computer Graphics Forum*, vol. 27, no. 6, pp. 1539–1556, Wiley Online Library, 2008.
- [SHA 08b] SHAPIRA L., SHAMIR A., COHEN-OR D., “Consistent mesh partitioning and skeletonisation using the shape diameter function”, *The Visual Computer*, vol. 24, no. 4, p. 249, January 2008.
- [SHI 91] SHINAGAWA Y., KUNII T.L., KERGOSIEN Y., “Surface coding based on Morse theory”, *IEEE Computer Graphics and Applications*, vol. 222, pp. 66–78, 1991.
- [SHI 07] SHILANE P., FUNKHOUSER T., “Distinctive regions of 3D surfaces”, *ACM Transactions on Graphics*, vol. 26, no. 2, ACM, 2007.
- [SHL 02] SHLAFMAN S., TAL A., KATZ S., “Metamorphosis of polyhedral surfaces using decomposition”, *Computer Graphics Forum*, vol. 21, no. 3, Wiley Online Library, 2002.
- [SIM 13] SIMARI P., DE FLORIANI L., IURICICH F. *et al.*, “Generalized extrinsic distortion and applications”, *Computers & Graphics*, vol. 37, no. 6, pp. 582–588, October 2013.
- [SIN 90] SINHA S.S., BESL P.J., “Principal patches—a viewpoint-invariant surface description”, *Proceedings IEEE International Conference on Robotics and Automation*, vol. 1, pp. 226–231, May 1990.
- [SIP 11] SIPIRAN I., BUSTOS B., “Harris 3D: A robust extension of the Harris operator for interest point detection on 3D meshes”, *The Visual Computer*, vol. 27, no. 11, pp. 963–976, 2011.
- [SKI 09] SKINNER M.M., WOOD B.A., HUBLIN J.-J., “Protostyloid expression at the enamel-dentine junction and enamel surface of mandibular molars of *Paranthropus robustus* and *Australopithecus africanus*”, *Journal of Human Evolution*, vol. 56, no. 1, pp. 76–85, 2009.
- [SON 14] SONG R., LIU Y., MARTIN R.R. *et al.*, “Mesh saliency via spectral processing”, *ACM Transactions on Graphics*, vol. 33, no. 1, pp. 6:1–6:17, ACM, 2014.
- [SOT 08] SOTOMAYOR J., GARCIA R., “Lines of curvature on surfaces, historical comments and recent developments”, *The São Paulo Journal of Mathematical Sciences*, vol. 2, no. 1, pp. 99–143, 2008.
- [SPI 99] SPIVAK M., *A Comprehensive Introduction to Differential Geometry*, Publish or Perish, Houston, Tex, 3rd edition, January 1999.
- [STE 71] STEFANELLI R., ROSENFELD A., “Some parallel thinning algorithms for digital pictures”, *Journal of ACM*, vol. 18, pp. 255–264, 1971.
- [STO 92] STOKELY E.M., WU S.Y., “Surface parametrization and curvature measurement of arbitrary 3D objects: Five practical methods”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 8, pp. 833–840, August 1992.
- [STY 04] STYLIANOU G., FARIN G., “Crest lines for surface segmentation and flattening”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 5, pp. 536–544, September 2004.
- [SUB 95] SUBSOL G., *Construction automatique d’atlas anatomiques morphométriques à partir d’images médicales tridimensionnelles*, PhD Thesis, Ecole Centrale de Paris, 1995.

- [SUB 97] SUBSOL G., ROBERTS N., DORAN M. *et al.*, “Automatic analysis of cerebral atrophy”, *Magnetic Resonance Imaging*, vol. 15, no. 8, pp. 917–927, 1997.
- [SUB 98] SUBSOL G., THIRION J.-P., AYACHE N., “A scheme for automatically building three-dimensional morphometric anatomical atlases: Application to a skull atlas”, *Medical Image Analysis*, vol. 2, no. 1, pp. 37–60, 1998.
- [SUB 99a] SUBSOL G., “Chapter 14 - crest lines for curve-based warping”, in TOGA W. (ed.), *Brain Warping*, pp. 241–262, Academic Press, San Diego, 1999.
- [SUB 99b] SUBSOL G., “Crest lines for curve-based warping”, in TOGA W. (ed.), *Brain Warping*, pp. 241–262, Academic Press, San Diego, 1999.
- [SUB 02] SUBSOL G., MAFART B., SILVESTRE A. *et al.*, “3D image processing for the study of the evolution of the shape of the human skull: Presentation of the tools and preliminary results”, in MAFART B., DELINGETTE H., SUBSOL G. (eds), *Three-Dimensional Imaging in Paleoanthropology and Prehistoric Archaeology*, BAR International Series 1049, pp. 37–45, Archeopress, Liège (Belgium), 2002.
- [SUB 09] SUBSOL G., BRAGA J., THACKERAY F. *et al.*, “Automatic crest lines extraction for 3D morphometry of fossil structures”, *Paleoanthropology Society Annual Meeting*, vol. 2009, A35, March 2009.
- [SUN 03] SUNDAR H., SILVER D., GAGVANI N. *et al.*, “Skeleton based shape matching and retrieval”, *Proceedings of the Shape Modeling International 2003, SMI'03*, p. 130, Washington, DC, USA, IEEE Computer Society, 2003.
- [TAK 14] TAKAYAMA K., JACOBSON A., KAVAN L. *et al.*, “A simple method for correcting facet orientations in polygon meshes based on ray casting (JCGT)”, *Journal of Computer Graphics Techniques (JCGT)*, vol. 3, no. 4, pp. 53–63, 2014.
- [TAK 16] TAKEZAWA M., IMAI T., SHIDA K. *et al.*, “Fabrication of freeform objects by principal strips”, *ACM Transactions on Graphics*, vol. 35, no. 6, pp. 225:1–225:12, ACM, November 2016.
- [TAU 95] TAUBIN G., “Estimating the tensor of curvature of a surface from a polyhedral approximation”, *5th International Conference on Computer Vision, 1995. Proceedings*, pp. 902–907, June 1995.
- [TAU 00] TAUBIN G., “Geometric signal processing on polygonal meshes”, *Eurographics 2000 - State of the The Art Reports*, Eurographics Association, 2000.
- [TEV 14] TEVS A., HUANG Q., WAND M. *et al.*, “Relating shapes via geometric symmetries and regularities”, *ACM Transactions on Graphics*, vol. 33, no. 4, pp. 119:1–12, 2014.
- [THE 15] THEOLOGOU P., PRATIKAKIS I., THEOHARIS T., “A comprehensive overview of methodologies and performance evaluation frameworks in 3D mesh segmentation”, *Computer Vision and Image Understanding*, vol. 135, pp. 49–82, 2015.
- [THI 95] THIRION J.-P., GOURDON A., “Computing the differential characteristics of isointensity surfaces”, *Computer Vision and Image Understanding*, vol. 61, no. 2, pp. 190–202, 1995.
- [THI 96a] THIRION J.-P., “The extremal mesh and the understanding of 3D surfaces”, *International Journal of Computer Vision*, vol. 19, no. 2, pp. 115–128, August 1996.

- [THI 96b] THIRION J.-P., GOURDON A., “The 3D marching lines algorithm”, *Graphical Models and Image Processing*, vol. 58, no. 6, pp. 503–509, 1996.
- [THI 96c] THIRION J.-P., SUBSOL G., DEAN D., “Cross validation of three inter-patients matching methods”, *Visualization in Biomedical Computing*, Lecture Notes in Computer Science, pp. 327–336, Springer, Berlin, Heidelberg, 1996.
- [TIE 17] TIERNY J., FAVELIER G., LEVINE J.A. *et al.*, “The topology toolKit”, *IEEE Transactions on Visualization and Computer Graphics*, Institute of Electrical and Electronics Engineers, October 2017.
- [TOM 13] TOMBARI F., SALTI S., DI STEFANO L., “Performance evaluation of 3D keypoint detectors”, *International Journal of Computer Vision*, vol. 102, nos 1–3, pp. 198–220, 2013.
- [TOP 05] TOPONOGOV V.A., *Differential Geometry of Curves and Surfaces: A Concise Guide*, Birkhauser Boston Inc, Boston, 2005.
- [TSU 17] TSUCHIE S., “Reconstruction of underlying surfaces from scanned data using lines of curvature”, *Computers & Graphics*, vol. 68, pp. 108–118, 2017.
- [VÁŠ 16] VÁŠA L., VANĚČEK P., PRANTL M. *et al.*, “Mesh statistics for robust curvature estimation”, *Computer Graphics Forum*, vol. 35, no. 5, pp. 271–280, 2016.
- [VÁŠ 17] VÁŠA L., KÜHNERT T., BRUNETT G., “Multivariate analysis of curvature estimators”, *Computer-Aided Design and Applications*, vol. 14, no. 1, pp. 58–69, Taylor & Francis, 2017.
- [VEM 93a] VEMURI B.C., MALLADI R., “Constructing intrinsic parameters with active models for invariant surface reconstruction”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 7, pp. 668–681, July 1993.
- [VEM 93b] VEMURI B.C., MALLADI R., “Intrinsic parameters for surface representation using deformable models”, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 2, pp. 614–623, March 1993.
- [VER 18] VERMA N., BOYER E., VERBEEK J., “FeaStNet: Feature-steered graph convolutions for 3D shape analysis”, *CVPR - IEEE Conference on Computer Vision & Pattern Recognition*, pp. 2598–2606, Salt Lake City, United States, IEEE, June 2018.
- [VID 11] VIDAL V., WOLF C., DUPONT F., “Robust feature line extraction on CAD triangular meshes”, *International Conference on Computer Graphics Theory and Applications*, Algarve, Portugal, March 2011.
- [WAN 09] WANG D., CLARK B., JIAO X., “An analysis and comparison of parameterization-based computation of differential quantities for discrete surfaces”, *Computer Aided Geometric Design*, vol. 26, no. 5, pp. 510–527, 2009.
- [WAT 01] WATANABE K., BELYAEV A.G., “Detection of salient curvature features on polygonal surfaces”, *Computer Graphics Forum*, vol. 20, no. 3, pp. 385–392, 2001.
- [WEA 55] WEATHERBURN C., *Differential Geometry of 3D*, vol. 1, Cambridge at the University Press, 1955.

- [WEY 04] WEYRICH T., PAULY M., KEISER R. *et al.*, “Post-processing of scanned 3D surface data”, *Proceedings of the First Eurographics Conference on Point-Based Graphics, SPBG’04*, pp. 85–94, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, 2004.
- [WIS 06] WISCHGOLL T., MEYER J., “Topological features in vector fields”, in BONNEAU G.-P., ERTL T., NIELSON G.M. (eds), *Scientific Visualization: The Visual Extraction of Knowledge from Data*, pp. 287–30, Berlin, Heidelberg, Springer Berlin Heidelberg, 2006.
- [WOO 21] WOODWARD A.S., “A new cave man from Rhodesia, South Africa”, *Nature*, vol. 108, pp. 371–372, November 1921.
- [WOR 03] WORKSHOP, “The Paris endoscopic classification of superficial neoplastic lesions: Esophagus, stomach, and colon: November 30 to December 1, 2002”, *Gastrointestinal Endoscopy*, vol. 58, no. 6, pp. S3–S43, December 2003.
- [WU 13] WU J., SHEN X., ZHU W. *et al.*, “Mesh saliency with global rarity”, *Graphical Model*, vol. 75, no. 5, pp. 255–264, Academic Press Professional, Inc., 2013.
- [XU 05] XU Z., XU G., SUN J.-G., “Convergence analysis of discrete differential geometry operators over surfaces”, in MARTIN R., BEZ H., SABIN M. (eds), *Mathematics of Surfaces XI: 11th IMA International Conference*, pp. 448–457, Loughborough, UK, Springer Berlin Heidelberg, Berlin, Heidelberg, September 5–7, 2005.
- [XU 06] XU G., “Convergence analysis of a discretization scheme for Gaussian curvature over triangular surfaces”, *Computer Aided Geometric Design*, vol. 23, no. 2, pp. 193–207, February 2006.
- [XU 09] XU Z., XU G., “Discrete schemes for Gaussian curvature and their convergence”, *Computers & Mathematics With Applications*, vol. 57, no. 7, pp. 1187–1195, 2009.
- [XU 13] XU G., “Consistent approximations of several geometric differential operators and their convergence”, *Applied Numerical Mathematics*, vol. 69, pp. 1–12, 2013.
- [YAN 06] YANG Y.-L., LAI Y.-K., HU S.-M. *et al.*, “Robust principal curvatures on multiple scales”, *Proceedings of the 4th Eurographics Symposium on Geometry Processing, SGP’06*, pp. 223–226, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, 2006.
- [YOS 02] YOSHIDA H., NÄPPI J., MACENEANEY P. *et al.*, “Computer-aided diagnosis scheme for detection of polyps at CT colonography”, *RadioGraphics*, vol. 22, no. 4, pp. 963–979, 2002.
- [YOS 05] YOSHIKAWA S., BELYAEV A.G., SEIDEL H.-P., “Fast and robust detection of crest lines on meshes”, *Proceedings of the ACM Symposium on Solid and Physical Modeling (SPM’05)*, pp. 227–232, 2005.
- [YOS 08] YOSHIKAWA S., BELYAEV A., YOKOTA H. *et al.*, “Fast, robust, and faithful methods for detecting crest lines on meshes”, *Computer Aided Geometric Design*, vol. 25, no. 8, pp. 545–560, 2008.
- [YU 07] YU J., YIN X., GU X. *et al.*, “Focal surfaces of discrete geometry”, *Proceedings of the 5th Eurographics Symposium on Geometry Processing, SGP’07*, pp. 23–32, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association, 2007.

- [YU 08] YU K., WU J., ZHUANG Y., “Skeleton-based recognition of chinese calligraphic character image”, *Advances in Multimedia Information Processing (PCM'08)*, vol. 5353 of *Lecture Notes in Computer Science*, pp. 228–237, Springer, 2008.
- [YUI 90] YUILLE A., LEYTON M., “3D symmetry-curvature duality theorems”, *Computer Vision, Graphics, and Image Processing*, vol. 52, no. 1, pp. 124–140, 1990.
- [ZAH 12] ZAHARESCU A., BOYER E., HORAUD R., “Keypoints and local descriptors of scalar functions on 2D manifolds”, *International Journal of Computer Vision*, vol. 100, no. 1, pp. 78–98, 2012.
- [ZHA 84] ZHANG T.Y., SUEN C.Y., “A fast parallel algorithm for thinning digital patterns”, *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, March 1984.
- [ZHA 06] ZHAO L., BOTHA C., BESCOS J. *et al.*, “Lines of curvature for polyp detection in virtual colonoscopy”, *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 885–892, September 2006.
- [ZHA 08a] ZHANG X., LI G., XIONG Y. *et al.*, “3D mesh segmentation using mean-shifted curvature”, in CHEN F., JÜTTLER B. (eds), *Advances in Geometric Modeling and Processing*, vol. 4975 of *Lecture Notes in Computer Science*, pp. 465–474, Springer Berlin Heidelberg, 2008.
- [ZHA 08b] ZHAO L., RAVESTEIJN V.F.V., BOTHA C.P. *et al.*, *Surface curvature line clustering for polyp detection in CT colonography*, The Eurographics Association, 2008.
- [ZHA 09] ZHANG X., CHE W., PAUL J.-C., “Computing lines of curvature for implicit surfaces”, *Computer Aided Geometric Design*, vol. 26, no. 9, pp. 923–940, 2009.
- [ZHE 17] ZHENG P., BELATON B., LIAO I.Y. *et al.*, “A functional pipeline framework for landmark identification on 3D surface extracted from volumetric data”, *PLOS ONE*, vol. 12, no. 11, pp. 1–25, Public Library of Science, 11 2017.
- [ZHO 04] ZHOU K., SYNDER J., GUO B. *et al.*, “Iso-charts: Stretch-driven mesh parameterization using spectral analysis”, *Proceedings of the Eurographics symposium on Geometry processing (SGP'04)*, pp. 45–54, 2004.
- [ZOL 98] ZOLLIKOFER C.P.E., PONCE DE LEÓN M.S., MARTIN R.D., “Computer-assisted paleoanthropology”, *Evolutionary Anthropology: Issues, News, and Reviews*, vol. 6, no. 2, pp. 41–54, December 1998.
- [ZOM 05] ZOMORODIAN A., CARLSSON G.E., “Computing persistent homology”, *Discrete & Computational Geometry*, vol. 33, no. 2, pp. 249–274, 2005.
- [ZOM 08] ZOMORODIAN A., CARLSSON G., “Localized homology”, *Computational Geometry: Theory and Applications*, vol. 41, no. 3, pp. 126–148, Elsevier Science Publishers B.V., November 2008.
- [ZUC 02] ZUCKERBERGER E., TAL A., SHLAFMAN S., “Polyhedral surface decomposition with applications”, *Computers & Graphics*, vol. 26, no. 5, pp. 733–743, Elsevier, 2002.



AQ1. Please check page number.

