Mathematical Foundations of Data Sciences

Gabriel Peyré
CNRS & DMA
École Normale Supérieure
gabriel.peyre@ens.fr
www.gpeyre.com
www.numerical-tours.com

November 2, 2017

Chapter 3

Linear Mesh Processing

This chapter exposes the basics of surface approximation with 3D meshes and the way to process such meshes with linear operators. In particular, it studies filtering on 3D meshes and explains how a Fourier theory can be built to analyze these filters.

3.1 Surface Discretization with Triangulated Mesh

3.1.1 Continuous Geometry of Surfaces

In this course, in order to simplify the mathematical description of surfaces, we consider only globally parameterized surfaces. We begin by considering surfaces embedded in euclidean space $\mathcal{M} \subset \mathbb{R}^k$.

Definition 1 (Parameterized surface)**.** *A parameterized surface is a mapping*

$$u \in \mathcal{D} \subset \mathbb{R}^2 \mapsto \varphi(u) \in \mathcal{M}.$$

Of course, most surfaces do not benefit from such a simple parameterization. For instance, a sphere should be split into two parts in order to be mapped on two disks $\mathcal{D}_1, \mathcal{D}_2$. These topological difficulties require the machinery of manifolds in order to incorporate a set of charts $\mathcal{D} = \{\mathcal{D}_i\}_i$ that overlap in a smooth manner. All the explanations of this course extend seamlessly to this multi-charts setting.

A curve is defined in parameter domain as a 1D mapping $t \in [0,1] \mapsto \gamma(t) \in \mathcal{D}$. This curve can be traced over the surface and its geometric realization is $\bar{\gamma}(t) \stackrel{\text{def.}}{=} \varphi(\gamma(t)) \in \mathcal{M}$. The computation of the length of γ in ambient k-dimensional space $\mathbb{R}^k$ follows the usual definition, but to do the computation over the parametric domain, one needs to use a local metric defined as follow.

Definition 2 (First fundamental form)**.** *For an embedded manifold $\mathcal{M} \subset \mathbb{R}^k$, the first fundamental form is*

$$I_\varphi = \left(\langle \frac{\partial \varphi}{\partial u_i}, \frac{\partial \varphi}{\partial u_j} \rangle \right)_{i,j=1,2}.$$

This local metric I_φ defines at each point the infinitesimal length of a curve as

$$L(\gamma) \stackrel{\text{def.}}{=} \int_0^1 \|\bar{\gamma}'(t)\| \mathrm{d}t = \int_0^1 \sqrt{\gamma'(t)^{\mathrm{T}} I_\varphi(\gamma(t)) \gamma'(t)} \mathrm{d}t.$$

This fundamental form is an intrinsic invariant that does not depends on how the surfaces is isometrically embedded in space (since the length depends only on this tensor field I_φ). In contrast, higher order differential quantities such as curvature might depend on the bending of the surface and are thus usually not intrinsic (with the notable exception of invariants such as the gaussian curvature). In this course, we restrict ourselves to first order quantities since we are mostly interested in lengths and the intrinsic study of surfaces.

Example 1 (Isometry and conformality). A surface $\mathcal{M}$ is locally isometric to the plane if $I_\varphi = \mathrm{Id}_2$. This is for instance the case for a cylinder. The mapping φ is said to be conformal if $I_\varphi(u) = \lambda(u)\mathrm{Id}_2$. It means that the length of a curve over the plane is only locally scaled when mapped to the surface. In particular, the angle of two interescting curves is the same over the parametric domain and over the surface. This is for instance the case for the stereographic mapping between the plane and a sphere.

3.1.2 Discretization of Surfaces with Triangulations

Mesh Data Structure A triangulated mesh is a discrete structure that can be used to approximate a surface embedded in Euclidean space $\mathbb{R}^k$. It is composed of a topological part $M = (V, E, F)$ and a geometrical realization $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$. It is important to make the distinction between these two parts since many algorithms rely only on geometry (point clouds processings such as dimension reduction) or on topology (such as compression).

The topology M of the mesh is composed of

– *Vertices* (0D): this is an abstract set of indices $V \simeq \{1, \ldots, n\}$.

– *Edges* (1D): this is a set of pair of vertices $E \subset V \times V$. This set is assumed to be symmetric

$$(i,j) \in E \quad \Longleftrightarrow \quad i \sim j \Leftrightarrow (j,i) \in E.$$

– *Faces* (2D): this is a collection of 3-tuples of vertices $F \subset V \times V \times V$, with the additional compatibility condition

$$(i,j,k) \in F \quad \Longrightarrow \quad (i,j), (j,k), (k,i) \in E.$$

We further assumer that there is no isolated edges

$$\forall\, (i,j) \in E, \quad \exists\, k,\ (i,j,k) \in F.$$

The set of edges can be stored in a symmetric matrix $A \in \mathbb{R}^{n \times n}$ such that $A_{ij} = 1$ if $(i,j) \in E$ and $A_{ij} = 0$ otherwise. This matrix is often stored as a sparse matrix since the number of edges is usually much smaller than n^2. The set of vertices and edges form a non-oriented graph $\mathcal{G} = (V, E)$. Faces are often stored as a matrix $A_F \in \{1, \ldots, n\}^{3 \times m}$ where m is the number of faces and a column $((A_F)_{i,1}, (A_F)_{i,2}, (A_F)_{i,3})$ stores the indices of a face. In a triangulation, the face matrix A_F allows to recover the edge incidence matrix A. The face data structure allows to really capture the 2D geometry of surfaces, which is not possible with graphs alone.

The geometric realization $\mathcal{M}$ is defined through a spacial localization of the vertices (for instance in 3D space)

$$\mathcal{V} \stackrel{\text{def.}}{=} \{x_i\ ;\ i \in V\} \subset \mathbb{R}^3.$$

This allows to define a piecewise linear mesh

$$\mathcal{F} \stackrel{\text{def.}}{=} \bigcup_{(i,j,k) \in F} \mathrm{Conv}(x_i, x_j, x_k) \subset \mathbb{R}^3,$$

where the convex envelop $\mathrm{Conv}(x, y, z)$ of three points is the Euclidean triangle generated by (x, y, z).

This piecewise linear realization $\mathcal{M}$ can be displayed as a 3D surface on a computer screen. This is performed through a perspective projection of the points and a linear interpolation of color and light inside the triangle. Figure 3.1 shows an example of 3D display, with a zoom on the faces of the mesh.

Adjacency Relationships From the basis topological information given by $M = (V, E, F)$, one can deduce several adjacency data-structures that are important to navigate over the triangulation.

Definition 3 (Vertex 1-ring)**.** *The vertex 1-ring of a vertex $i \in V$ is*

$$V_i \stackrel{\text{def.}}{=} \{j \in V\ ;\ (i,j) \in E\} \subset V.\text{`} \tag{3.1}$$

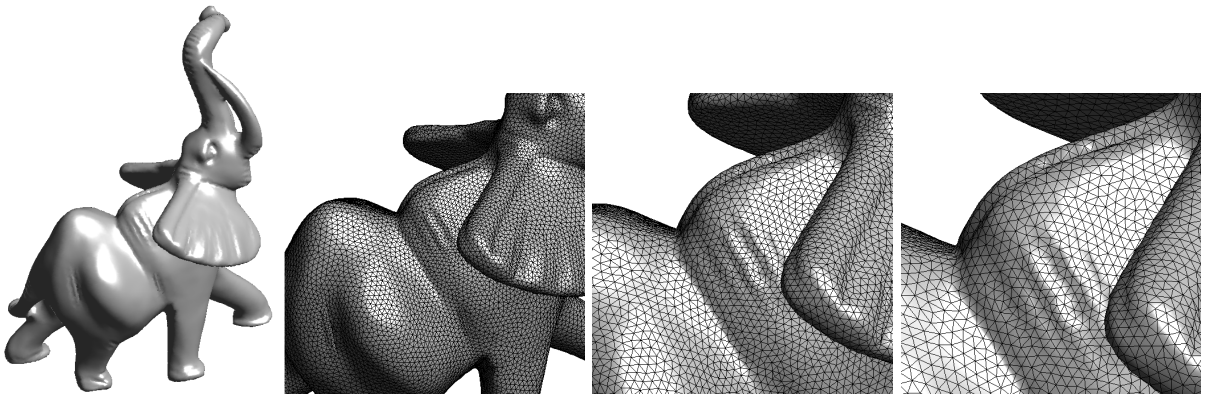

Figure 3.1: Example of display of a 3D mesh.

The s-ring is defined by induction as

$$\forall\, s > 1, \quad V_i^{(s)} = \left\{ j \in V \;;\; (k,j) \in E \quad \textit{and} \quad k \in V_i^{(s-1)} \right\}. \tag{3.2}$$

Definition 4 (Face 1-ring)**.** *The face 1-ring of a vertex $i \in V$ is*

$$F_i \stackrel{\text{def.}}{=} \{(i,j,k) \in F \;;\; i,j \in V\} \subset F.$$

The geometrical realization of a vertex 1-ring is

$$\mathcal{V}_i = \bigcup_{(i,j,k) \in V_i} \mathrm{Conv}(x_i, x_j, x_k).$$

A triangulated mesh is a manifold mesh if all the rings $\mathcal{V}_i$ for $i \in V$ are homeomorphic to either a disk (for interior vertices) or to a half disk (for boundary vertices). This ensures that the geometrical mesh really has the topology of a 2D surface embedded in $\mathbb{R}^3$ (possibly with boundaries). In particular, it implies that there is at most two faces connected to each edge

$$\forall\, (i,j) \in E, \quad \#\{k \;;\; (i,j,k) \in F\} \leqslant 2.$$

As an application of these local rings, one can compute a normal at each point using a simple rule

$$\forall\, f = (i,j,k) \in F, \quad \overrightarrow{n_f} \stackrel{\text{def.}}{=} \frac{(x_j - x_i) \wedge (x_k - x_i)}{\|(x_j - x_i) \wedge (x_k - x_i)\|}.$$

and where

$$\forall\, i \in V, \quad \overrightarrow{n_i} \stackrel{\text{def.}}{=} \frac{\sum_{f \in F_i} \overrightarrow{n_f}}{\| \sum_{f \in F_i} \overrightarrow{n_f} \|}.$$

These normals are used to define for instance a light intensity $I(i) = \max(\langle n_i, \ell(i), , \rangle 0)$, where $\ell(i)$ is the incident light. In practice one uses a infinite light source $\ell(i) = \ell$ =constant or a local spot located at position $s \in \mathbb{R}^3$ through $\ell(i) = (v_i - s)/\|v_i - s\|$. This light intensity is interpolated on the whole mesh during display.

3.2 Linear Mesh Processing

The light intensity I is a particular example of a function defined at each vertex of the mesh. Mesh processing is intended to process such functions and we thus define carefully vector spaces and operators on meshes.

3.2.1 Functions on a Mesh

In this course, a function is a discrete set of values defined at each vertex location.

Definition 5 (Linear space on a mesh). *A function on a mesh is a mapping $f \in \ell^2(\mathcal{V}) \simeq \ell^2(V) \simeq \mathbb{R}^n$ and can be viewed equivalently as*

$$f : \left\{ \begin{array}{ccc} \mathcal{V} & \longrightarrow & \mathbb{R} \\ x_i & \longmapsto & f(x_i) \end{array} \right. \iff f : \left\{ \begin{array}{ccc} V & \longrightarrow & \mathbb{R} \\ i & \longmapsto & f_i \end{array} \right. \iff f = (f_i)_{i \in V} \in \mathbb{R}^n.$$

The linear space of the functions on a mesh is equipped with an Hilbert space structure that allows to quantify approximation error and compute projections of functions.

Definition 6 (Inner product and norm). *One defines the following inner product and norm for vector $f, g \in \mathbb{R}^n$*

$$\langle f, g \rangle \stackrel{\text{def.}}{=} \sum_{i \in V} f_i g_i \quad \textit{and} \quad \|f\|^2 = \langle f, f \rangle.$$

In order to modify (process) functions on a mesh (such as a light intensity I), this course considers only linear operations that are defined through a large matrix.

Definition 7 (Linear operator A). *A linear operator A is defined as*

$$A : \ell^2(V) \to \ell^2(V) \iff A = (a_{ij})_{i,j \in V} \in \mathbb{R}^{n \times n} \textit{ (matrix).}$$

and operate on a function f as follow

$$(Af)(x_i) = \sum_{j \in V} a_{ij} f(x_j) \iff (Af)_i = \sum_{j \in V} a_{ij} f_j.$$

Example 2. If the coordinates of the point of a mesh are written $x_i = (x_i^1, x_i^2, x_i^3) \in \mathbb{R}^3$, then the X-coordinate defines a function $f : i \in V \mapsto x_i^1 \in \mathbb{R}$. A geometric mesh $\mathcal{M}$ is thus 3 functions defined on M.

Mesh processing is the task of modifying functions $f \in \ell^2(V)$. For instance, one can denoise a mesh $\mathcal{M}$ as 3 functions on M. The usual strategy applies a linear operator $f \mapsto Af$. Sometimes, A can computed from M only (for instance for compression) but most of the times it requires both M and $\mathcal{M}$.

3.2.2 Local Operators

In most applications, one can not store and manipulate a full matrix $A \in \mathbb{R}^{n \times n}$. Furthermore, one is usually interested in exploiting the local redundancies that exist in most usual functions $f \in \mathbb{R}^n$ defined on a mesh. This is why we restrict our attention to local operators that can be conveniently stored as sparse matrices (the zeros are not kept in memory).

Definition 8 (Local operator). *A local operator $W \in \mathbb{R}^{n \times n}$ satisfies $w_{ij} = 0$ if $(i,j) \notin E$.*

$$(Wf)_i = \sum_{(i,j) \in E} w_{ij} f_j.$$

A particularly important class of local operators are local smoothings (also called filterings) that perform a local weighted sum around each vertex of the mesh. For this averaging to be consistent, we define a normalized operator $\tilde{W}$ whose set of weights sum to one.

Definition 9 (Local averaging operator). *A local normalized averaging is $\tilde{W} = (\tilde{w}_{ij})_{i,j \in V} \geqslant 0$ where*

$$\forall\, (i,j) \in E, \quad \tilde{w}_{ij} = \frac{w_{ij}}{\sum_{(i,j) \in E} w_{ij}}.$$

It can be equivalently expressed in matrix form as

$$\tilde{W} = D^{-1} W \quad \textit{with} \quad D = \mathrm{diag}_i(d_i) \quad \textit{where} \quad d_i = \sum_{(i,j) \in E} w_{ij}.$$

The smoothing property corresponds to $\tilde{W}1 = 1$ which means that the unit vector is an eigenvector of W with eigenvalue 1.

Example 3. In practice, we use three popular kinds of averaging operators.

– *Combinatorial weights:* they depends only on the topology (V, E) of the vertex graph

$$\forall\,(i,j) \in E, \quad w_{ij} = 1.$$

– *Distance weights:* they depends both on the geometry and the topology of the mesh, but do not require faces information,

$$\forall\,(i,j) \in E, \quad w_{ij} = \frac{1}{\|x_j - x_i\|^2}.$$

– *Conformal weights:* they depends on the full geometrical realization of the 3D mesh since they require the face information

$$\forall\,(i,j) \in E, \quad w_{ij} = \cot(\alpha_{ij}) + \cot(\beta_{ij}). \tag{3.3}$$

Figure 3.2 shows the geometrical meaning of the angles α_{ij} and β_{ij}

$$\alpha_{ij} = \angle(x_i, x_j, x_{k_1}) \qquad \text{and} \qquad \beta_{ij} = \angle(x_i, x_j, x_{k_2}),$$

where $(i,j,k_1) \in F$ and $(i,j,k_2) \in F$ are the two faces adjacent to edge $(i,j) \in E$. We will see in the next section the explanation of these celebrated cotangent weights.

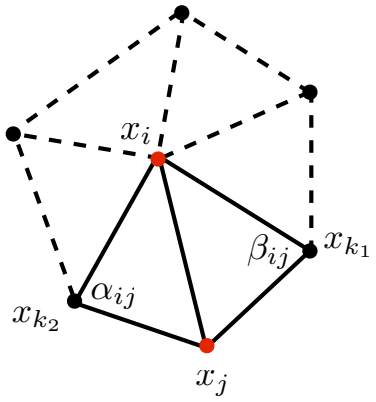

Figure 3.2: One ring around a vertex i, together with the geometrical angles α_{ij} and β_{ij} used to compute the conformal weights.

One can use iteratively a smoothing in order to further filter a function on a mesh. The resulting vectors $\tilde{W}f, \tilde{W}^2, \ldots, \tilde{W}^k f$ are increasingly smoothed version of f. Figure 3.3 shows an example of such iterations applied to the three coordinates of mesh. The sharp features of the mesh tend to disappear during iterations. We will make this statement more precise in the following, by studying the convergence of these iterations.

3.2.3 Approximating Integrals on a Mesh

Before investigating algebraically the properties of smoothing operators, one should be careful about what are these discrete operators really approximating. In order for the derivation to be simple, we make computation for a planar triangulation M of a mesh $\mathcal{M} \subset \mathbb{R}^2$.

In the continuous domain, filtering is defined through integration of functions over the mesh. In order to descretize integrals, one needs to define a partition of the plane into small cells centered around a vertex or an edge.

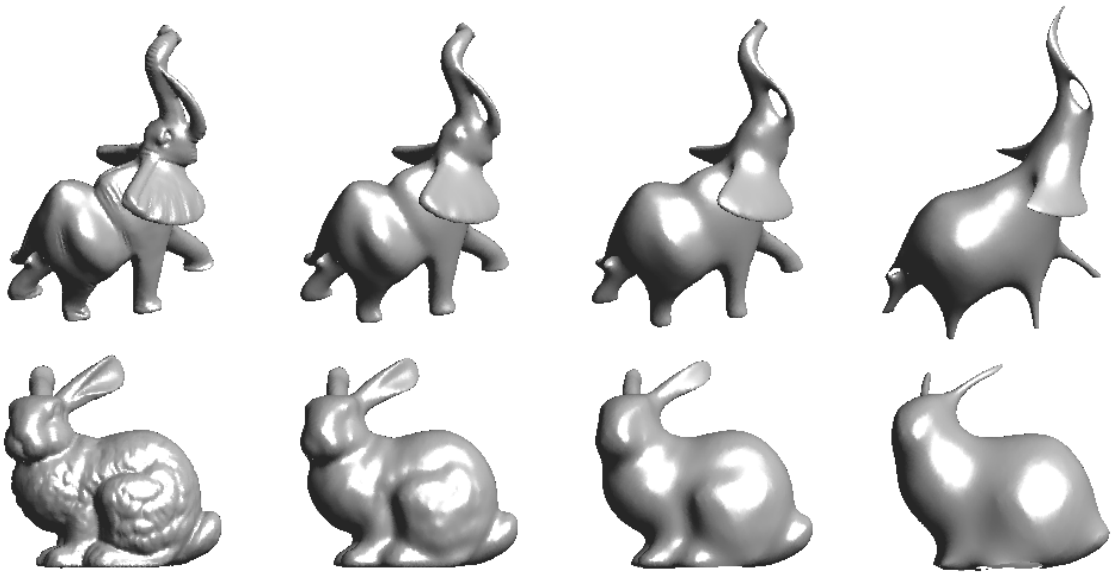

Figure 3.3: Examples of iterative smoothing of a 3D mesh.

Definition 10 (Vertices Voronoi)**.** *The Voronoi diagram associated to the vertices is*

$$\forall\, i \in V, \quad E_i = \{x \in \mathcal{M}\ ;\ \forall\, j \neq i, \|x - x_i\| \leqslant \|x - x_j\|\}$$

Definition 11 (Edges Voronoi)**.** *The Voronoi diagram associated to the edges is*

$$\forall\, e = (i,j) \in E, \quad E_e = \{x \in \mathcal{M}\ ;\ \forall\, e' \neq e, d(x,e) \leqslant d(x,e')\}$$

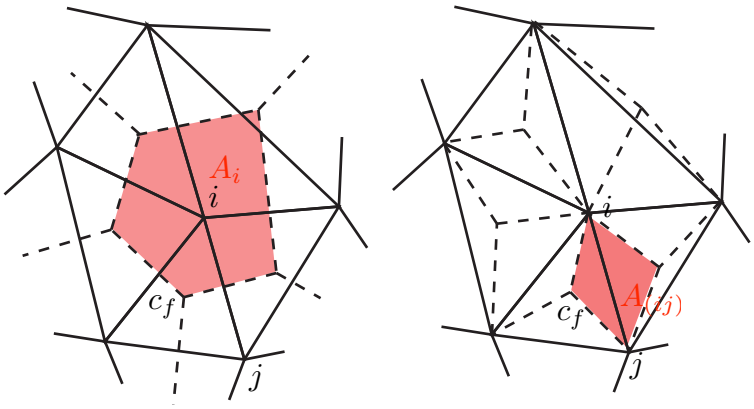

Figure 3.4: Left: vertex Voronoi cell, right: delaunay Voronoi cell. The point c_f is the orthocenter of a face $f = (i,j,k)$.

These Voronoi cells indeed form a partition of the mesh

$$\mathcal{M} = \bigcup_{i \in V} E_i = \bigcup_{e \in E} E_e.$$

The following theorem gives the formula for the area of these cells.

Theorem 3 (Voronoi area formulas). *For all $e = (i, j) \in E$, $\forall\, i \in V$, one has*

$$A_e = Area(E_e) = \frac{1}{2}\|x_i - x_j\|^2 \left(\cot(\alpha_{ij}) + \cot(\beta_{ij})\right)$$

$$A_i = Area(E_i) = \frac{1}{2}\sum_{j \in N_i} A_{(ij)}.$$

With these areas, one can approximate integrals on vertices and edges using

$$\int_{\mathcal{M}} f(x)\mathrm{d}x \approx \sum_{i \in V} A_i\, f(x_i) \approx \sum_{e=(i,j)\in E} A_e\, f([x_i, x_j]).$$

Of particular interest is the approximation of the so-called Dirichelet energy $\int_{\mathcal{M}} \|\nabla_x f\|^2 \mathrm{d}x$. In order to compute it on a triangular mesh, one can use a finite difference approximation of the gradient of a function at the point $x_{ij} = (x_i + x_j)/2$ along an edge (i, j)

$$\langle \nabla_{x_{ij}} f,\, \frac{x_i - x_j}{\|x_i - x_j\|} \rangle \approx \frac{f(x_i) - f(x_j)}{\|x_i - x_j\|}.$$

This leads to the following approximation of the Dirichlet energy

$$\int_{\mathcal{M}} \|\nabla_x f\|^2 \mathrm{d}x \approx \sum_{(i,j)\in E} A_{(i,j)} \langle \nabla_{x_{ij}} f,\, \frac{x_i - x_j}{\|x_i - x_j\|} \rangle^2 \approx \sum_{(i,j)\in E} A_{(i,j)}\, \frac{|f(x_j) - f(x_i)|^2}{\|x_j - x_i\|^2} \tag{3.4}$$

$$= \sum_{(i,j)\in E} w_{ij} |f(x_j) - f(x_i)|^2 \qquad \text{where} \qquad w_{ij} = \cot(\alpha_{ij}) + \cot(\beta_{ij}). \tag{3.5}$$

This discrete formulation shows that the correct weights to approximate the Dirichlet energy are the cotangent one, already introduced in equation (3.3).

3.2.4 Example on a Regular Grid

A regular grid is an uniform discretization with n points of $[0, 1)$ (in 1D) or $[0, 1)^2$ (in 2D). One usually assumes periodic boundary conditions, which means that each side of the square is associated with its opposite.

Since the geometry of a regular grid is invariant under translation, local averaging operators can be computed as convolution on $D = (\mathbb{Z}/p\mathbb{Z})^d$ where $n = p^d$ for d the dimension of the domain ($d = 1$ or $d = 2$)

$$\forall\, i \in D, \quad \tilde{W} f(i) = \sum_{k \in D} f(k)\tilde{w}(i - k),$$

where the operation $+$ and $-$ should be computed modulo p and $\tilde{w}(k) = \tilde{W}(0, k)$ is the convolution kernel.

Example 4 (Averaging). The uniform averaging filter is defined as

$$\tilde{W} f(i) = \frac{1}{|N|} \sum_{k \in N} f(i + k),$$

where N is the set of neighbors of the point 0 and $|N| = 2^d$. In this case, in dimension 1, $\tilde{w} = (1, 0, 1)/2$, where this notation assumes that $\tilde{w}$ is centered at the point 0.

In order to study translation invariant operators like local filtering, one needs to use the discrete Fourier transform that diagonalizes these operators.

Definition 12 (Discrete Fourier transform). *The 1D discrete Fourier transform $\Phi(f) \in \mathbb{C}^n$ of the vector $f \in \mathbb{C}^n$*

$$\Phi(f)(\omega) = \hat{f}(\omega) \stackrel{\text{def.}}{=} \frac{1}{n} \sum_k f_k e^{\frac{2\mathrm{i}\pi}{n} k\omega}.$$

A similar definition can be given for the 2D discrete Fourier transform. The main property of the Fourier transform is the following diagonalization result.

Theorem 4 (Convolution and Fourier). *For any vector f, one has*

$$\Phi(\tilde{W}^k f) = \Phi(\tilde{w} * \ldots * \tilde{w} * f) \quad \Longrightarrow \quad \Phi(\tilde{W}^k f)(\omega) = \widehat{\tilde{w}}(\omega)^k \, \hat{f}(\omega).$$

The main interest of this tools is that $\Phi(f)$ can be computed in $O(n \log(n))$ operations with the FFT algorithm. Using the following theorem, it gives an alternative expression of a local filtering. This expression in the Fourier domain can be used to speed up the computation of $\tilde{w} * f$ if $\tilde{w}$ has a lot of non zero entries (which is not the case in our setting of local operators). It is also useful to analyze theoretically the behavior of iterated filterings.

Theorem 5 (Convergence). *For any function f defined on a regular grid in 1D or 2D, one has*

$$\tilde{W}^k f \stackrel{k \to +\infty}{\longrightarrow} \frac{1}{|V|} \sum_{i \in V} f_i$$

This Fourier theory can only be developed for domains that have a group structure that enables translation invariant filtering. In particular, it does not carry over easily to an arbitrary surface. In the remaining, we define a corresponding theory for graphs and triangulated surfaces using the eigenvector of Laplacian operators. This Fourier transform on meshes enables the analysis of the convergence of many filtering schemes.

3.2.5 Gradients and Laplacians on Meshes

Gradient operator A gradient operator defines directional derivatives on a triangulation. It maps functions defined on vertices to functions defined on the set of oriented edges

$$\bar{E} \stackrel{\text{def.}}{=} \{(i,j) \in E \,;\, i > j\}.$$

Definition 13 (Gradient). *Given a local averaging W, the gradient operator G is defined as*

$$\forall\, (i,j) \in E,\ i < j, \quad (Gf)_{(i,j)} \stackrel{\text{def.}}{=} \sqrt{w_{ij}}(f_j - f_i) \in \mathbb{R}.$$

This mapping can be viewed equivalently as

$$\begin{array}{lll} G : \ell^2(V) \longrightarrow \ell^2(E), & \text{or} & G : \mathbb{R}^n \longrightarrow \mathbb{R}^p \quad \text{where} \quad p = |E|, \\ & \text{or} & G \in \mathbb{R}^{n \times p} \quad \text{(a matrix)}. \end{array}$$

The value of $(Gf)_e$ for an edge $e = (i,j)$ can be thought as a derivative along direction $\overrightarrow{x_i x_j}$.

Example 5. For the local averaging based on square distances, one has

$$w_{ij} = \|x_i - x_j\|^{-2}, \quad (Gf)_{(i,j)} = \frac{f(x_j) - f(x_i)}{\|x_i - x_j\|}.$$

which is exactly the finite difference discretization of a directional derivative.

One a regular grid, one can note that

– Gf discretizes $\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)^{\mathrm{T}}$.

– $G^{\mathrm{T}} v$ discretizes $\div(v) = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y}$.

Laplacian Operator A Laplacian operator is a discrete version of a second order derivative operator.

Definition 14 (Laplacian)**.** *Given a local averaging W, the Laplacian operator D is defined as*

$$L \overset{\text{def.}}{=} D - W, \quad \textit{where} \quad D = \operatorname{diag}_i(d_i), \quad \textit{with} \quad d_i = \sum_j w_{ij}.$$

In the remaining, we also make use of normalized operators, which have an unit diagonal.

Definition 15 (Normalized Laplacian)**.** *The normalized Laplacian is defined as*

$$\tilde{L} \overset{\text{def.}}{=} D^{-1/2} L D^{-1/2} = \mathrm{Id}_n - D^{-1/2} W D^{1/2} = \mathrm{Id}_n - D^{1/2} \tilde{W} D^{-1/2}.$$

This normalized Laplacian correspond to the weighted graph Laplacian used in graph theory, see for instance [13].

Remark 1. One can note that
– Laplacians are symmetric operators $L, \tilde{L} \in \mathbb{R}^{n \times n}$.
– L acts like a (second order) derivative since $L1 = 0$.
– in contrast, the normalized Laplacian is not a real derivative since $\tilde{L}1 \neq 0$ in general.

The main interest of the gradient operator is that it factorizes the Laplacian as follow.

Theorem 6 (Laplacian factorization)**.** *One has*

$$L = G^{\mathrm{T}} G \quad \textit{and} \quad \tilde{L} = (G D^{-1/2})^{\mathrm{T}} (G D^{-1/2}).$$

This theorem proves in particular that L and $\tilde{L}$ are symmetric positive definite operators. The inner product defined by the Laplacian can be expressed as an energy summed over all the edges of the mesh

$$\langle Lf,\, f \rangle = \|Gf\|^2 = \sum_{(i,j) \in E} w_{ij} \|f_i - f_j\|^2.$$

In the particular case of the cotangent weights introduced in equation (3.3), one can see that the Laplacian norm $\langle Lf,\, f \rangle$ is exactly the finite differences approximation of the continuous Dirichlet energy $\int_{\mathcal{M}} |\nabla_x f| \mathrm{d}x$ derived in equation (3.5). This is why these cotangent weights are the best choice to compute a Laplacian that truly approximates the continuous Laplace Beltrami operator (see definition 16).

A similar expression is derived for the normalized laplacian

$$\langle \tilde{L} f,\, f \rangle = \|G D^{-1/2} f\|^2 = \sum_{(i,j) \in E} w_{ij} \left\| \frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right\|^2.$$

Of particular interest for the study of filtering on meshes is the behavior of the spectrum of the Laplacian. We can first study its kernel.

Theorem 7 (Kernel of the Laplacian)**.** *If M is connected, then*

$$\textit{ker}(L) = \textit{span}(1) \quad \textit{and} \quad \textit{ker}(L) = \textit{span}(D^{1/2}).$$

3.2.6 Examples in 1D and 2D

In 1D, all local weights are equivalent since the points are equi-spaced. The corresponding Laplacian is a convolution that can be written as

$$(Lf)_i = \frac{1}{h^2} (2 f_i - f_{i+1} - f_{i-1}) = \frac{1}{h^2} f * (-1, 2, 1),$$

where it is important to remember that the notation $(-1,2,1)$ means that the vector is centered around 0.

This discrete 1D Laplacian is the finite difference approximation of the continuous Laplacian on the torus $\mathcal{T}$ of the segment $[0,1)$ modulo 1. Up to a minus sign, this Laplacian is just the second order derivative

$$L \overset{h\to 0}{\longrightarrow} -\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}(x_i)$$

One should be careful with our notation that consider positive semi-definite Laplacian, that have the opposite sign with respect to second order derivative operators (which are definite negative).

The gradient operator corresponds to a discretization of the first order derivative $f \mapsto f'$ (which is anti symmetric). The continuous counterpart of the factorization $L = G^{\mathrm{T}}G$ is the integration by part formula on the torus

$$\int_{\mathcal{T}} f''(x)g(x)\mathrm{d}x = -\int_{\mathcal{T}} -f(x)g'(x)\mathrm{d}x \quad \Longrightarrow \quad \int_{\mathcal{T}} f''(x)f(x)\mathrm{d}x = -\int_{\mathcal{T}} |f'(x)|^2 \leqslant 0.$$

The discrete Laplacian on a 2D grid can also be written as a 2D convolution

$$(Lf)_i = \frac{1}{h^2}\left(4f_i - f_{j_1} - f_{j_2} - f_{j_3} - f_{j_4}\right) = \frac{1}{h^2} f * \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

where $\{j_k\}_k$ are the four neighbors of the point i. This operator is the finite difference approximation to the continuous 2D Laplacian

$$L \overset{h\to 0}{\longrightarrow} -\frac{\partial^2 f}{\partial x^2}(x_i) - \frac{\partial^2 f}{\partial y^2}(x_i) = -\Delta f(x_i).$$

The factorization $Lf = G^{\mathrm{T}}Gf$ corresponds to the decomposition $\Delta f = \div(\nabla f)$.

3.2.7 Example of a Parametric Surface

We recall that a parameterized surface is a mapping $u \in \mathcal{D} \subset \mathbb{R}^2 \mapsto \varphi(u) \in \mathcal{M}$. Whereas the continuous Laplacian is simple to define on the plane using partial derivatives, its definition on a surface requires the intervention of an arbitrary parameterization φ which makes its expression cumbersome.

Definition 16 (Laplace-Beltrami)**.** *The Laplace-Beltrami operator on a parametric surface $\mathcal{M}$ is defined as*

$$\sqrt{g}\Delta_{\mathcal{M}} \overset{\text{def.}}{=} \frac{\partial}{\partial u_1}\left(\frac{g_{22}}{\sqrt{g}}\frac{\partial}{\partial u_1} - \frac{g_{12}}{\sqrt{g}}\frac{\partial}{\partial u_2}\right) + \frac{\partial}{\partial u_2}\left(\frac{g_{11}}{\sqrt{g}}\frac{\partial}{\partial u_2} - \frac{g_{12}}{\sqrt{g}}\frac{\partial}{\partial u_1}\right)$$

where $g = \det(I_\varphi)$ and $I_\varphi = (g_{ij})_{i,j=1,2}$.

The Laplacian is however an intrinsic operator that does not depends on the chosen parameterization, as shown by the following approximation theorem.

Remark 2 (Laplacian using averaging)*.*

$$\Delta_{\mathcal{M}} f(x) = \lim_{h\to 0} \frac{1}{|B_h(x)|}\int_{y\in\mathcal{M}} f(y)\mathrm{d}y \quad \text{where} \quad B_h(x) = \{y \,;\, d_{\mathcal{M}}(x,y) \leqslant h\}$$

where $d_{\mathcal{M}}$ is the geodesic distance on $\mathcal{M}$ and $h = \max_{(i,j)\in E} \|x_i - x_j\|$ is the discretization precision.

3.3 Diffusion and Regularization on Surfaces

3.3.1 Heat Diffusion

The main linear PDE for regularization of functions is the heat equation that governs the isotropic diffusion of the values of a function in time.

Definition 17 (Heat diffusion). $\forall t > 0$, *one defines* $F_t : M \to \mathbb{R}$ *solving*

$$\frac{\partial F_t}{\partial t} = -D^{-1}LF_t = -(\mathrm{Id}_n - \tilde{W})F_t \qquad \text{and} \qquad \forall i \in V,\ F_0(i) = f(i)$$

In order to compute numerically the solution of this PDE, one can fix a time step $\delta > 0$ and use an explicit discretization in time $\bar{F}_k$ as $F_0 = f$ and

$$\frac{1}{\delta}\left(\bar{F}_{k+1} - \bar{F}_k\right) = -D^{-1}L\bar{F}_k \quad \Longrightarrow \quad \bar{F}_{k+1} = \bar{F}_k - \delta D^{-1}L\bar{F}_k = (\mathrm{Id} - \delta)\bar{F}_k + \delta\tilde{W}\bar{F}_k. \tag{3.6}$$

If δ is small enough, one hopes that the discrete solution $\bar{F}_k$ is close to the continuous time solution F_t for $t = \delta k$. This is indeed the case as proven later in these notes.

Remark 3. In order for this scheme to be stable, one needs $\delta < 1$. This is be proven later using the extension of Fourier theory to meshes.

Remark 4. If $\delta = 1$, then the discretization of the Heat equation corresponds to iterative smoothing since $\bar{F}_k = \tilde{W}^k f$. In this case stability is not guaranteed but only pathological meshes give unstable filtering (see theorem 15).

Instead of using the explicit discretization in time (3.6), one can use an implicit scheme which compute an approximate solution $\tilde{F}_k$ at step k by solving

$$\frac{1}{\delta}\left(\tilde{F}_{k+1} - \tilde{F}_k\right) = -D^{-1}L\tilde{F}_{k+1} \quad \Longrightarrow \quad ((\delta+1)\mathrm{Id}_n - \delta\tilde{W})\tilde{F}_{k+1} = \tilde{F}_k. \tag{3.7}$$

Computing $\tilde{F}_k$ requires the solution of a sparse linear system at each step k. The implicit scheme (3.7) is thus computationally more involved than the explicit scheme (3.6). We will however see later that the implicit scheme is always stable for any value of $\delta \leqslant 1$.

Example 6 (Mesh smoothing). In order to smooth a mesh whose points are $x_i = (x_i^1, x_i^2, x_i^3)$, one can perform a heat diffusion for each component $f_i = (x_i^k), k = 1, 2, 3$. Figure 3.5 shows an example of such a smoothing.

In practice, mesh smoothing is used to denoise a function $f = f_0 + \sigma g$ where $g \in \mathbb{R}^n$ is a realization of a gaussian white noise (each entry $g(i)$ are independent and follow a gaussian law with unit variance). The difficult task it to find an optimal stopping time t to minimize $\|F_t - f_0\|$, which is not available since one does not know f_0. For uniformly smooth surfaces, the theory predicts that a linear filtering such as the heat equation requires a stopping time proportional to the noise level σ. This is however false for more complex surfaces such as the one used in computer graphics. In these case, alternate non linear diffusions such as non-linear PDE or wavelet thresholding usually perform better, see [28] for an overview of these methods in image processing.

Other differential equations. One can solve other partial differential equations involving the Laplacian over a 3D mesh $M = (V, E, F)$. For instance, one can consider the wave equation, which defines, for all $t > 0$, a vector $F_t \in \ell^2(V)$ as the solution of

$$\frac{\partial^2 F_t}{\partial t^2} = -D^{-1}LF_t \qquad \text{and} \qquad \begin{cases} F_0 = f \in \mathbb{R}^n, \\ \frac{\mathrm{d}}{\mathrm{d}t}F_0 = g \in \mathbb{R}^n, \end{cases} \tag{3.8}$$

In order to compute numerically the solution of this PDE, one can fix a time step $\delta > 0$ and use an explicit discretization in time $\bar{F}_k$ as $F_0 = f$, $F_1 = F_0 + \delta g$ and for $k > 1$

$$\frac{1}{\delta^2}\left(\bar{F}_{k+1} + \bar{F}_{k-1} - 2\bar{F}_k\right) = -D^{-1}L\bar{F}_k \quad \Longrightarrow \quad \bar{F}_{k+1} = 2\bar{F}_k - \bar{F}_{k-1} - \delta^2 D^{-1}L\bar{F}_k.$$

Figure 3.6 shows examples of the resolution of the wave equation on 3D meshes.

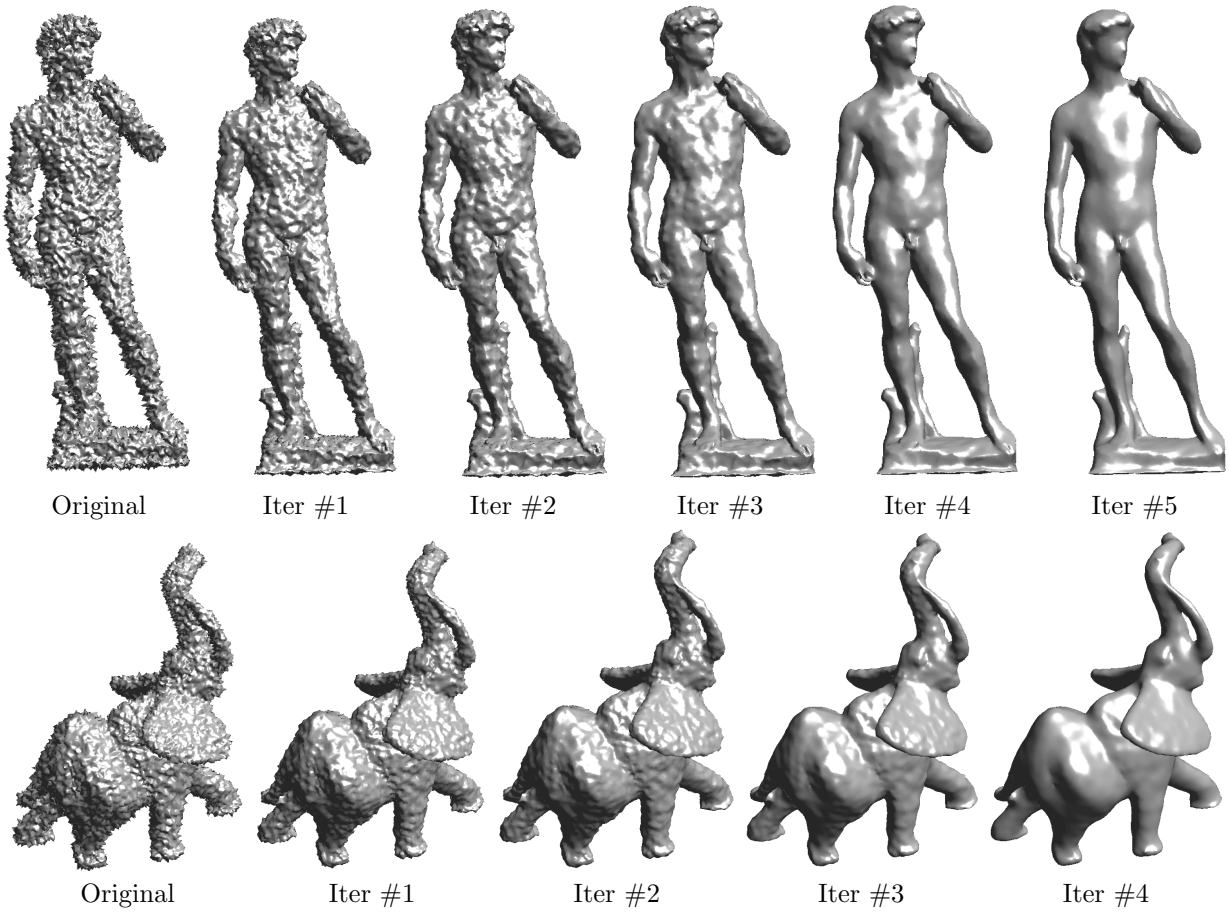

Figure 3.5: Examples of mesh denoising with the heat equation.

3.3.2 Spectral Decomposition

In order to better understand the behavior of linear smoothing on meshes, one needs to study the spectral content of Laplacian operators. This leads to the definition of a Fourier theory for meshes. The decomposition $\tilde{L} = (GD^{-1/2})^{\mathrm{T}}(GD^{-1/2})$ of the Laplacian implies that it is a positive semi-definite operator. One can thus introduce the following orthogonal factorization.

Theorem 8 (Eigen-decomposition of the Laplacian)**.** *It exists a matrix* U, $\quad U^{\mathrm{T}}U = \mathrm{Id}_n$ *such that*

$$\tilde{L} = U\Lambda U^{\mathrm{T}} \qquad \textit{where} \qquad \Lambda = \mathrm{diag}_\omega(\lambda_\omega), \qquad \lambda_1 \leqslant \ldots \leqslant \lambda_n.$$

The eigenvalues λ_ω correspond to a frequency index that ranks the eigenvectors u_ω of $U = (u_\omega)_\omega$. One can first state some bounds on these eigenvalues.

Theorem 9 (Spectral bounds)**.** $\forall\, i,\ \lambda_i \in [0, 2]$ *and*
– *If* M *is connected then* $0 = \lambda_1 < \lambda_2$.
– $\lambda_n = 2$ *if and only if* M *is 2-colorable.*

We recall the definition of a colorable graph next.

Definition 18 (Colorable graph)**.** *A graph* (V, E) *is* k*-colorable if it exist a mapping* $f : V \to \{1, \ldots, k\}$ *such that*

$$\forall\, (i, j) \in E, \quad f(i) \neq f(j).$$

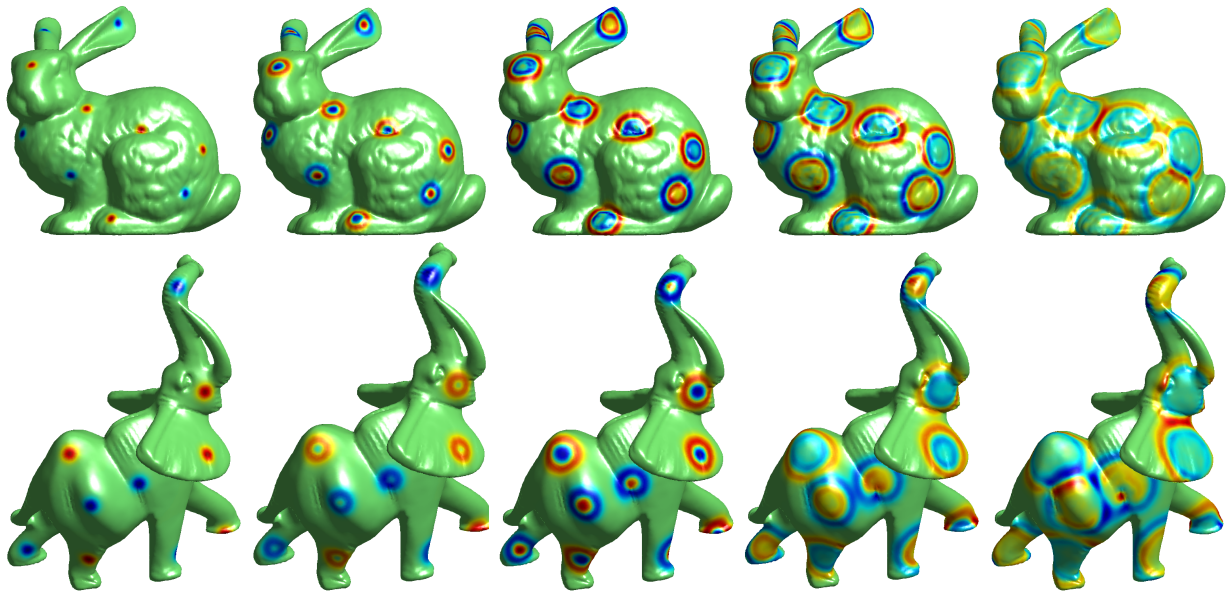

Figure 3.6: Example of evolution of the wave equation on 3D mesh. The initial condition f is a superposition of small positive and negative gaussians.

A 2-colorable graph is also called bi-partite. A 2-colorable mesh is pathological for filtering since one can split the set of vertices into two parts without inner connexions. The filtering process can oscillate by exchanging values between these sets, thus never converging.

The orthogonal eigen-basis $U = (u_\omega)_\omega$ is an orthogonal basis of the space $\mathbb{R}^n \simeq \ell^2(V)$, which can be written as

$$u_\omega : \left\{ \begin{array}{ccc} V & \longrightarrow & \mathbb{R} \\ i & \longmapsto & u_\omega(x_i) \end{array} \right.$$

The orthogonality means that $\langle u_\omega, u_{\omega'} \rangle = \delta_\omega^{\omega'}$. This basis allows to compute an orthogonal decomposition of any functions f

$$\forall\, f \in \ell^2(V), \quad f = \sum_\omega \langle f, u_\omega \rangle u_\omega.$$

Having such a tool allows to split a function f in elementary contributions $\langle f, u_\omega \rangle$ with a control in the energy because of orthogonality

$$\|f\|^2 = \sum_\omega |\langle f, u_\omega \rangle|^2.$$

Figure 3.7 shows some examples of eigenfunctions depicted using color ranging from blue (negative values of the eigenfunction) to red (positive values). One can see that these functions are oscillating, in a way similar to the traditional Fourier basis. In some sense (made more precise latter), this basis is the extension of the Fourier basis to meshes. A function u_ω corresponding to a large spectral value λ_ω is highly oscillating and corresponds thus intuitively to a high frequency atom.

Extracting numerically eigenvectors from a large matrix is a difficult problem. If the matrix is sparse, a method of choice consists in using iterative powers of a shifted version of the laplacian. One starts from a random initial vector v_0 and iterates

$$v_{k+1} = \frac{w_{k+1}}{\|w_{k+1}\|} \qquad \text{where} \qquad w_{k+1} = (\tilde{L} - \lambda \mathrm{Id}_n)^{-1} v_k. \tag{3.9}$$

These iterates converges to the eigenvectors corresponding to the eigenvalue the closest to λ, as staten in the following theorem.

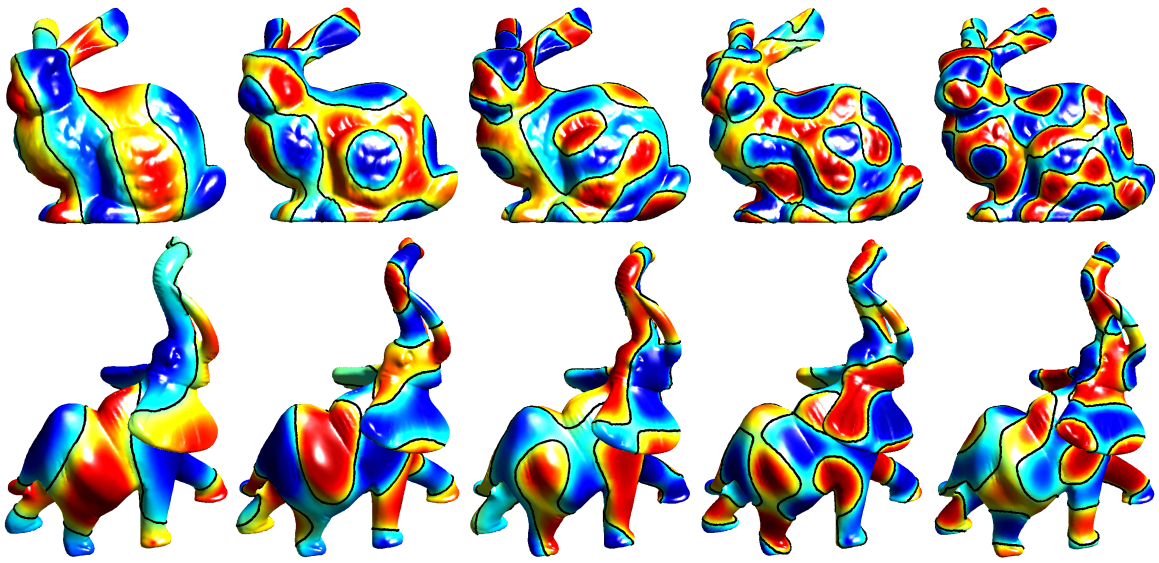

Figure 3.7: Examples of eigenvectors u_ω of the Laplacian $\tilde{L}$. The blue colors indicated negative values, red colors positive ones. The black curve is the 0 level set of the eigenvector.

Theorem 10 (Inverse iterations). *For a given shift λ, lets denote*

$$\omega^\star = \operatorname*{argmin}_\omega |\lambda - \lambda_\omega| \qquad \textit{and} \qquad \omega^+ = \operatorname*{argmin}_{\omega \neq \omega^\star} |\lambda - \lambda_\omega|$$

If $|\lambda - \lambda_{\omega^\star}| < |\lambda - \lambda_{\omega^+}|$, then

$$v_k \overset{k\to+\infty}{\longrightarrow} u_{\omega^\star} \quad \textit{and} \quad \langle L v_k, v_k \rangle \overset{k\to+\infty}{\longrightarrow} \lambda_{\omega^\star}.$$

The speed of convergence of these inverse iterations is governed by the conditioning of $(\tilde{L} - \lambda \mathrm{Id}_n)^{-1}$ since

$$\|v_k - u_{\omega^\star}\| \leqslant C \rho(\lambda)^k \qquad \text{where} \qquad \rho(\lambda) \overset{\text{def.}}{=} \frac{|\lambda - \lambda_{\omega^\star}|}{|\lambda - \lambda_{\omega^+}|} < 1.$$

The smallest $\rho(\lambda)$ is, the faster the method converges.

In order to compute an iteration (3.9) of the method, one needs to solve a sparse linear system $A w_{k+1} = v_k$ whith $A = \tilde{L} - \lambda \mathrm{Id}_n$. In order to do so, one can use a direct method such as LU factorization. The advantage of such an approach is that the factorization is computed once for all and can be re-used to solve very quickly at each step k. These factorization are however quite slow to compute especially for large matrices. For large problems, one can solve this linear system using an iterative algorithm such as conjugate gradient. These iterative method are attractive for sparse matrices, but a fast convergence requires $1/\rho(\lambda)$, the conditioning of $\tilde{L} - \lambda \mathrm{Id}_n$ to be not large, with is contradictory with the constraint for iterations 3.9 to converge fast.

3.3.3 Spectral Theory on a Regular Grid

In the particular case of a 1D or 2D lattice, the eigenfunctions defined earlier correspond exactly to the Fourier basis used in the discrete Fourier transform.

Theorem 11 (Spectrum in 1D). *For a 1D regular lattice,*

$$u_\omega(k) = \frac{1}{\sqrt{n}} \exp\left(\frac{2\mathrm{i}\pi}{n} k\omega\right) \qquad \textit{and} \qquad \lambda_\omega = 4 \sin^2\left(\frac{2\pi}{n}\omega\right).$$

Theorem 12 (Spectrum in 2D). *For a 2D regular lattice, $n = n_1 n_2$, $\omega = (\omega_1, \omega_2)$*

$$u_\omega(k) = \frac{1}{\sqrt{n}} \exp\left(\frac{2\mathrm{i}\pi}{n}\langle k, \omega\rangle\right) \qquad \text{and} \qquad \lambda_\omega = 4\left(\sin^2\left(\frac{2\pi}{n_1}\omega_1\right) + \sin^2\left(\frac{2\pi}{n_2}\omega_2\right)\right).$$

As already mentioned, on a mesh, the eigenvectors of $\tilde{L}$ correspond to a extension of the Fourier basis to meshes. The definition of the Fourier transform on meshes requires a little care since a diagonal normalization by D is used as defined next.

Definition 19 (Manifold-Fourier transform). *For $f \in \ell^2(V)$,*

$$\Phi(f)(\omega) = \hat{f}(\omega) \overset{\text{def.}}{=} \langle D^{1/2} f, u_\omega\rangle \quad \Longleftrightarrow \quad \Phi(f) = \hat{f} = U^{\mathrm{T}} D^{1/2}.$$

where $(u_\omega)_\omega$ are the eigenvectors of $\tilde{L}$.

One can note that there is still a degree of freedom in designing this Fourier transform since one can use any local weighting (for instance combinatorial, distance or conformal). Depending on the application, one might need to use weights depending only on the topology of the mesh (combinatorial for mesh compression).

A major theoretical interest of this Fourier transform is that it diagonalizes local averaging operators.

Theorem 13 (Spectral smoothing). *One has $\Phi\tilde{W}\Phi^{-1} = \mathrm{Id}_n - \Lambda$ and thus for any function f*

$$\widehat{\tilde{W} f}(\omega) = (1 - \lambda_\omega)\hat{f}(\omega)$$

This diagonalization allows to prove the convergence of iterative smoothing.

Theorem 14 (Convergence of iterated smoothing). *If $\lambda_n < 2$ (i.e. M is not 2-colorable), then for any function f*

$$\tilde{W}^k f \overset{k\to+\infty}{\longrightarrow} \frac{1}{n}\sum_{i\in V} f_i.$$

3.3.4 Spectral Resolution of the Heat Diffusion

Recall that the heat diffusion is defined as

$$\forall\, t > 0, \quad \frac{\partial F_t}{\partial t} = -D^{-1} L F_t = -(\mathrm{Id}_n - \tilde{W}) F_t$$

Using the manifold Fourier expansion $\hat{F}_t \overset{\text{def.}}{=} U^{\mathrm{T}} D^{1/2} F_t$, this differential equation can be re-written as

$$\frac{\partial \hat{F}_t(\omega)}{\partial t} = -\lambda_\omega \hat{F}_t(\omega) \quad \Longrightarrow \quad \hat{F}_t(\omega) = \exp(-\lambda_\omega t)\hat{f}(\omega). \tag{3.10}$$

This allows to study the convergence of the continuous heat equation.

Theorem 15 (Convergence of heat equation). *If $\mathcal{M}$ is connected,*

$$F_t \overset{t\to+\infty}{\longrightarrow} \frac{1}{n}\sum_{i\in V} f_i.$$

Recall that the heat equation is discretized using the following explicit and implicit schemes, equations (3.6) and (3.7)

$$\begin{cases} \bar{F}_k = (1-\delta)\bar{F}_k + \delta\tilde{W}\bar{F}_k, \\ ((1+\delta)\mathrm{Id}_n - \delta\tilde{W})\tilde{F}_{k+1} = \tilde{F}_k. \end{cases}$$

These filtering iterations can be re-written over the Fourier domain as

$$\begin{cases} \widehat{\bar{F}_{k+1}}(\omega) = (1 - \delta\lambda_\omega)\widehat{\bar{F}_k}(\omega), \\ \widehat{\tilde{F}_{k+1}}(\omega) = \frac{1}{(1+\delta\lambda_\omega)}\widehat{\bar{F}_k}(\omega). \end{cases}$$

This allows to state the stability and convergence of the finite difference discretization.

Theorem 16 (Convergence of discretization). *The explicit scheme is stable if $\delta < 1$. The implicit scheme is always stable. One has*

$$\begin{cases} \bar{F}_{t/\delta} \xrightarrow{\delta\to 0} F_t, \\ \tilde{F}_{t/\delta} \xrightarrow{\delta\to 0} F_t. \end{cases}$$

with the restriction that for the explicit scheme, the mesh must not be 2-colorable.

Other Differential Equations. The manifold Fourier transform can also be used to solve the wave equation (3.8) since

$$\frac{\partial^2 \hat{F}_t(\omega)}{\partial t^2} = -\lambda_\omega \hat{F}_t(\omega) \quad \Longrightarrow \quad \hat{F}_t(\omega) = \cos(\sqrt{\lambda_\omega} t)\hat{f}(\omega) + \frac{1}{\sqrt{\lambda_\omega}} \sin(\sqrt{\lambda_\omega} t)\hat{g}(\omega).$$

3.3.5 Quadratic Regularization

Instead of using a PDE for regularization, one can try to find a new function that is both close to the original one f and that is smooth in a certain sense. This leads to the notion of quadratic regularization, where one uses a Laplacian as a smoothness prior on the recovered function.

Definition 20 (Quadratic regularizer). *For $t > 0$, one defines*

$$F_t^q = \operatorname*{argmin}_{g\in\mathbb{R}^n} \|f - g\|^2 + t\|\tilde{G}g\|^2 \qquad \textit{where} \qquad \tilde{G} = GD^{-1/2}.$$

This optimization replaces $f \in \ell^2(V)$ by $F_t^{\mathrm{q}} \in \ell^2(V)$ with small gradients. This optimization can be found in closed form by inverting a sparse linear system.

Theorem 17 (Solution of quadratic regularization). *F_t^q is unique and*

$$F_t^q = (\mathrm{Id}_n + t\tilde{L})^{-1} f.$$

Over the Fourier domain, this inversion reads

$$\hat{F}_t^{\mathrm{q}}(\omega) = \frac{1}{1 + t\lambda_\omega} \hat{f}(\omega).$$

This corresponds to an attenuation of the high frequency content of f, in a way very similar to equation (3.10).

Once again, similarly to the heat equation, the spectral expression of the quadratic regularizer allows to study its convergence for large t.

Theorem 18 (Convergence of quadratic regularization). *If $\mathcal{M}$ is connected,*

$$F_t^q \xrightarrow{t\to+\infty} \frac{1}{n}\sum_{i\in V} f_i.$$

3.3.6 Application to Mesh Compression

We have shown how the Fourier basis on meshes can be used to compute in a diagonal fashion filtering, heat diffusion and quadratic regularization. This Fourier transform is however of little interest in practice, since the original filterings (or finite difference approximation of the heat equation) are usually faster to compute directly than over the Fourier domain. The Fourier transform is thus mainly of theoretical interest in these cases since it allows to prove convergence results.

Another class of applications makes use of an orthogonal expansion such as the Fourier one to perform mesh compression. This section shows how to compute a linear M-term approximation in this Fourier basis and to do mesh compression. We refer to the survey [1] for more advanced non-linear mesh compression methods.

The orthogonal basis $U = (u_\omega)_\omega$ of $\ell^2(V) \simeq \mathbb{R}^n$, where $\tilde{L} = U\Lambda U^{\mathrm{T}}$ allows to define a linear approximation as followed.

Definition 21 (Linear M-term approximation). *For any $M > 0$, the linear M-term approximation of f is*

$$f = \sum_{\omega=1}^{n} \langle f,\, u_\omega \rangle u_\omega \quad \overset{M\text{-term approx.}}{\Longrightarrow} \quad f_M \overset{\text{def.}}{=} \sum_{\omega=1}^{M} \langle f,\, u_\omega \rangle u_\omega.$$

The quality of the approximation is measured using the error decay, which can in turn be estimated using the removed coefficients

$$E(M) \overset{\text{def.}}{=} \|f - f_M\|^2 = \sum_{\omega > M} |\langle f,\, u_\omega \rangle|^2.$$

A good orthogonal basis U is a basis for which $E(M)$ decays fast on the signals of interest. Equivalently, a fast decay of E with M corresponds to a fast decay of $|\langle f,\, u_\omega \rangle|$ for large ω. Figure 3.8 shows the decay of the Fourier spectrum for two different functions defined on a 3D mesh. The smooth function (left in the figure) exhibits a fast decay of its spectrum, meaning that it can be well approximated with only a few Fourier coefficients.

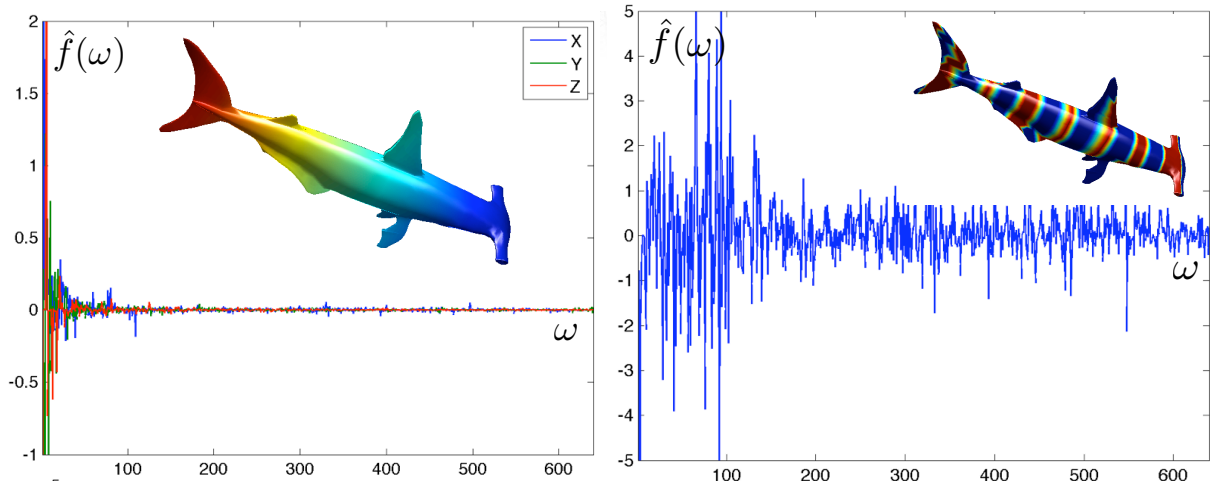

Figure 3.8: Examples of Fourier spectrum for a smooth and a non-smooth function.

We recall that the Fourier atoms

$$\forall\, \omega \in \mathbb{Z}, \quad u_\omega(x) = \frac{1}{\sqrt{2\pi}} e^{\mathrm{i}\omega x}$$

are the eigenvectors of the compact, symmetric, semi-definite negative operator $f \mapsto f''$ (that should be defined on the Hilbert space of twice Sobolev derivable functions). This set of function is also an Hilbert basis of the space $\mathrm{L}^2(\mathbb{R}/(2\pi\mathbb{Z}))$ of 2π-periodic square integrable functions and a Fourier coefficient is $\hat{f}(\omega) \overset{\text{def.}}{=} \langle f,\, u_\omega \rangle$.

Approximation theory studies this linear error decay for classical functional spaces. One can for instance study the Fourier expansion over euclidean spaces.

Theorem 19 (Fourier in 1D). *If f is C^α regular on $\mathbb{R}/(2\pi\mathbb{Z})$,*

$$\left|\hat{f}(\omega)\right| \leqslant \|f^{(\alpha)}\|_\infty |\omega|^{-\alpha}.$$

This result can be proven with a simple integration by parts. A slightly more difficult result shows that the linear approximation error decays like $M^{-\alpha}$.

Theorem 20 (Fourier approximation). *If f is C^α on $\mathbb{R}/(2\pi\mathbb{Z})$, then it exist $C > 0$ such that*

$$\sum_\omega |\omega|^{2\alpha} |\langle f, u_\omega\rangle|^2 < +\infty \quad \Longrightarrow \quad E(M) \leqslant CM^{-\alpha}.$$

This kind of results can be extended to continuous surfaces thanks to the continuous Laplacian. We suppose that $\mathcal{M}$ is a surface parameterized by φ, and a function $f = \varphi \circ \bar{f}$ is defined on it. By definition, this function f is C^α if $\bar{f}$ is C^α in euclidean space. For a compact surface $\mathcal{M}$, the Laplace-Beltrami operator $\Delta_\mathcal{M}$ is symmetric (for the inner product on the surface), is negative semi-definite and has a discrete spectrum $\Delta_\mathcal{M} u_\omega = -\lambda_\omega u_\omega$ for $\omega \in \mathbb{N}$. The functions $\{u_\omega\}_\omega$ are an orthogonal basis for function of finite energy on the surface $\mathrm{L}^2(\mathcal{M})$. The inner product of an arbitrary smooth function $f \in \mathrm{C}^\alpha(\mathcal{M})$ can be bounded using integration by parts

$$\langle f, u_\omega\rangle = \frac{1}{\lambda_\omega^k} \langle \Delta_\mathcal{M}^k f, u_\omega\rangle \quad \Longrightarrow \quad |\langle f, u_\omega\rangle| \leqslant \frac{\|f\|_{\mathrm{C}^\alpha}}{\lambda_\omega^{\alpha/2}}.$$

This proves the efficiency of the Fourier basis on surfaces to approximate smooth functions.

When computing the M-term approximation f_M of f one removes the small amplitude Fourier coefficients of the orthogonal expansion of f. Figure 3.9 shows some examples of mesh approximation where one retains an increasing number of Fourier coefficients. Mesh compression is only a step further, since one also need to code the remaining coefficients. This requires first quantifying the coefficients up to some finite precision and then binary code these coefficients into a file.

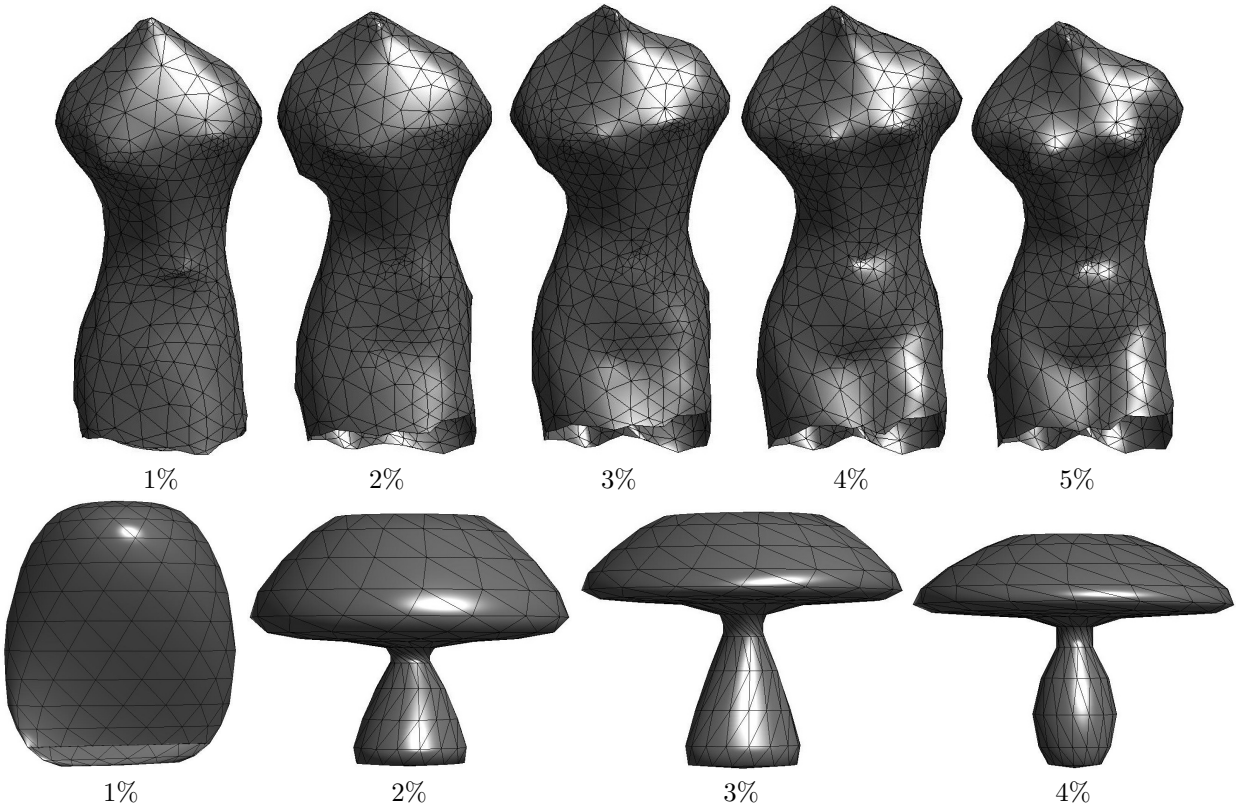

Figure 3.9: Examples of spectral mesh compression.

3.3.7 Application to Mesh Parameterization

This section is restricted to the study of meshes that can be globally parameterized on a plane. It means that they are topologically equivalent to a 2D disk. More complex meshes should be first segmented in cells that are equivalent to a disk.

A parameterization of a continuous surface $\mathcal{M}$ is a bijection

$$\psi : \mathcal{M} \longrightarrow \mathcal{D} \subset \mathbb{R}^2.$$

A similar definition applies to a discrete mesh where one computes a 2D position $\psi(i)$ for all the vertices $i \in V$ and then interpolates linearly the mapping to the whole piecewise linear geometric mesh. This section explains the basics of linear methods for mesh parameterization. We refer to various surveys [22, 42] for more details on mesh parameterization.

Usually, a 2D mesh is computed from range scanning or artistic modeling, so it does not come with such a parameterization. In order to perform texture mapping or more general mesh deformations, it is however important to use such a parameterization. Since many bijections are possible to layout the mesh in 2D, the mapping ψ has to satisfy additional smoothness assumptions. Classically, one requires that each coordinate of ψ has a vanishing Laplacian (it is thus harmonic) outside a set of constrained vertices that enforce boundary conditions.

More precisely, $\psi = (\psi_1, \psi_2)$ is the solution of

$$\begin{cases} \forall\, i \notin \partial\mathcal{M}, & (L\psi_1)(i) = (L\psi_2)(i) = 0 \\ \forall\, i \in \partial\mathcal{M}, & \psi(i) = \psi^0(i) \in \partial\mathcal{D}, \end{cases}$$

where $\partial\mathcal{M}$ is the boundary of the mesh, which consists in vertices whose face ring is not homeomorphic to a disk but rather to a half disk. This formulation requires the solution of two sparse linear systems (one for each coordinate of ψ).

The boundary condition $\psi^0(i)$ for $i \in \partial\mathcal{M}$ describes a 1D piecewise linear curve in the plane, that is fixed by the user. In the following, we will see that this curve should be convex for the parameterization to be bijective.

Remark 5. For such an harmonic parameterization, each point is the average of its neighbors since

$$\forall\, i, \quad \psi(i) = \frac{1}{\sum_j w_{ij}} \sum_{(i,j)\in E} w_{i,j}\psi(j).$$

The powerful feature of this linear parameterization method is that it can be proven to produce a valid (bijective) parameterization as long as the constrained position (boundary values of ψ) are along a convex curve.

Theorem 21 (Tutte theorem)**.** *If $\forall\,(i,j) \in E,\ w_{ij} > 0$, and if $\partial\mathcal{D}$ is a convex curve, then ψ is a bijection.*

Figure 3.10 shows several examples of parameterizations. One is free to use any laplacian (combinatorial, distance or conformal) as long as it produces positive weights. There is a issue with the conformal weights, which can be negative if the mesh contains obtuse triangles. In practice however it leads to the best results. The efficiency of a parameterization can be measured by some amount of distortion induced by the planar mapping. Linear methods cannot hope to cope with large isoperimetric distortions (for instance large extrusions in the mesh) since harmonicity leads to clustering of vertices.

3.3.8 Application to Mesh Flattening

One of the difficulty with linear parameterization methods is that they require to set up the positions of the vertices along the boundary of the mesh. In order to let the boundary free to evolve and find some optimal shape, one can replace the fixed point constraint by a global constraints of unit variance as follow

$$\min_{\psi_1,\psi_2\in\mathbb{R}^n} \|\tilde{G}\psi_1\|^2 + \|\tilde{G}\psi_2\|^2 \qquad \text{with} \qquad \begin{cases} \|\psi_i\| = 1, \\ \langle \psi_1,\, \psi_2 \rangle = 0, \\ \langle \psi_i,\, 1 \rangle = 0. \end{cases}$$

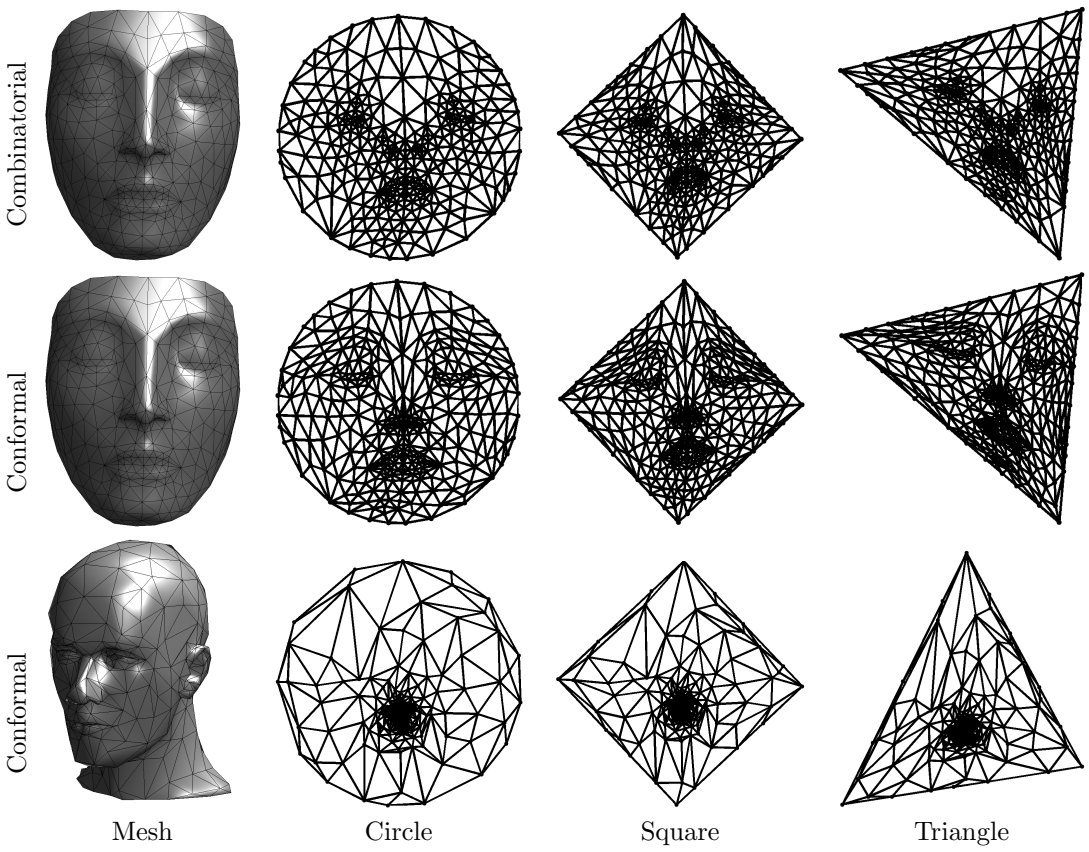

Figure 3.10: Examples of mesh parameterizations.

This optimization problem also has a simple global solution using eigenvectors of the Laplacian.

Theorem 22 (Mesh flattening solution)**.** *The mesh flattening solution is given by*

$$Span(\psi_1, \psi 2) = Span(u_1, u_2) \quad where \quad \tilde{L} = U \Lambda U^{\mathrm{T}}.$$

In order to compute this flattening, one thus needs to extract 2 eigenvectors from a sparse matrix. Note however that, in contrast to linear parameterization schemes, this flattening is not ensured to be bijective. Figure 3.11 shows that for meshes with large distortion, this flattening indeed leads to wrong parameterizations.

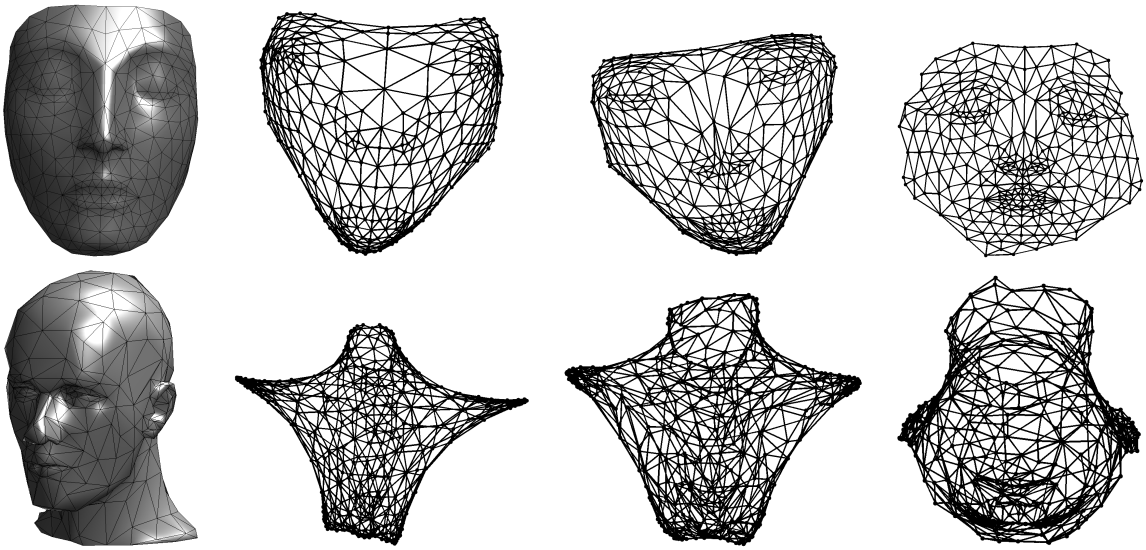

Figure 3.11: Examples of mesh flattening.

Bibliography

[1] P. Alliez and C. Gotsman. Recent advances in compression of 3d meshes. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 3–26. Springer Verlag, 2005.

[2] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene. Recent advances in remeshing of surfaces. In *AIM@SHAPE repport*. 2005.

[3] Amir Beck. *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB*. SIAM, 2014.

[4] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] E. Candès and D. Donoho. New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities. *Commun. on Pure and Appl. Math.*, 57(2):219–266, 2004.

[7] E. J. Candès. The restricted isometry property and its implications for compressed sensing. *Compte Rendus de l'Académie des Sciences*, Serie I(346):589–592, 2006.

[8] E. J. Candès, L. Demanet, D. L. Donoho, and L. Ying. Fast discrete curvelet transforms. *SIAM Multiscale Modeling and Simulation*, 5:861–899, 2005.

[9] A. Chambolle. An algorithm for total variation minimization and applications. *J. Math. Imaging Vis.*, 20:89–97, 2004.

[10] Antonin Chambolle, Vicent Caselles, Daniel Cremers, Matteo Novaga, and Thomas Pock. An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227, 2010.

[11] Antonin Chambolle and Thomas Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.

[12] S.S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

[13] F. R. K. Chung. Spectral graph theory. *Regional Conference Series in Mathematics, American Mathematical Society*, 92:1–212, 1997.

[14] Philippe G Ciarlet. Introduction à l'analyse numérique matricielle et à l'optimisation. 1982.

[15] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM Multiscale Modeling and Simulation*, 4(4), 2005.

[16] P. Schroeder et al. D. Zorin. Subdivision surfaces in character animation. In *Course notes at SIGGRAPH 2000*, July 2000.

[17] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. on Pure and Appl. Math.*, 57:1413–1541, 2004.

[18] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. *J. Fourier Anal. Appl.*, 4(3):245–267, 1998.

[19] D. Donoho and I. Johnstone. Ideal spatial adaptation via wavelet shrinkage. *Biometrika*, 81:425–455, Dec 1994.

[20] Heinz Werner Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.

[21] M. Figueiredo and R. Nowak. An EM Algorithm for Wavelet-Based Image Restoration. *IEEE Trans. Image Proc.*, 12(8):906–916, 2003.

[22] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pages 157–186. Springer Verlag, 2005.

[23] Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

[24] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In Alyn Rockwood, editor, *Proceedings of the Conference on Computer Graphics (Siggraph99)*, pages 325–334. ACM Press, August8–13 1999.

[25] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. In *Proceedings of the Computer Graphics Conference 2000 (SIGGRAPH-00)*, pages 271–278, New York, July 23–28 2000. ACMPress.

[26] L. Kobbelt. $\sqrt{3}$ subdivision. In Sheila Hoffmeyer, editor, *Proc. of SIGGRAPH'00*, pages 103–112, New York, July 23–28 2000. ACMPress.

[27] M. Lounsbery, T. D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Trans. Graph.*, 16(1):34–73, 1997.

[28] S. Mallat. *A Wavelet Tour of Signal Processing, 3rd edition.* Academic Press, San Diego, 2009.

[29] Stephane Mallat. *A wavelet tour of signal processing: the sparse way.* Academic press, 2008.

[30] D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and associated variational problems. *Commun. on Pure and Appl. Math.*, 42:577–685, 1989.

[31] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[32] Gabriel Peyré. *L'algèbre discrète de la transformée de Fourier.* Ellipses, 2004.

[33] Gabriel Peyré and Marco Cuturi. Computational optimal transport. 2017.

[34] J. Portilla, V. Strela, M.J. Wainwright, and Simoncelli E.P. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. Image Proc.*, 12(11):1338–1351, November 2003.

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

[36] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60(1-4):259–268, 1992.

[37] Filippo Santambrogio. Optimal transport for applied mathematicians. *Birkäuser, NY*, 2015.

[38] Otmar Scherzer, Markus Grasmair, Harald Grossauer, Markus Haltmeier, Frank Lenzen, and L Sirovich. *Variational methods in imaging*. Springer, 2009.

[39] P. Schröder and W. Sweldens. Spherical Wavelets: Efficiently Representing Functions on the Sphere. In *Proc. of SIGGRAPH 95*, pages 161–172, 1995.

[40] P. Schröder and W. Sweldens. Spherical wavelets: Texture processing. In P. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*. Springer Verlag, Wien, New York, August 1995.

[41] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[42] A. Sheffer, E. Praun, and K. Rose. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171, 2006.

[43] Jean-Luc Starck, Fionn Murtagh, and Jalal Fadili. *Sparse image and signal processing: Wavelets and related geometric multiscale analysis*. Cambridge university press, 2015.

[44] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computation Harmonic Analysis*, 3(2):186–200, 1996.

[45] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, 1997.