Geometry Processing
Exercise 4

2013

Hand in: 7. 11. 2013

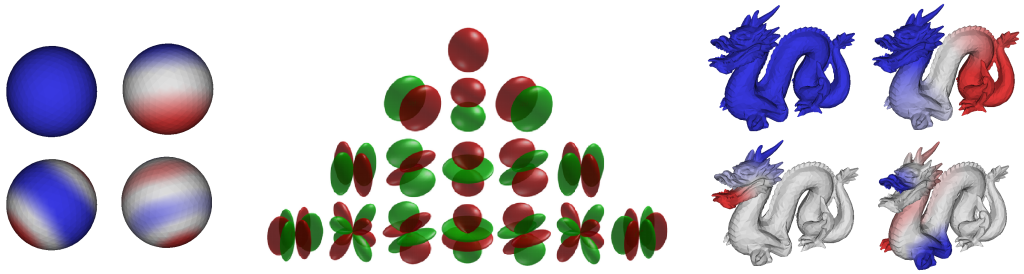

Figure 1: Left: four spherical harmonics, computed as eigenvectors of a discrete Laplacian. Middle: the first sixteen spherical harmonics, taken from Wikipedia. Right: manifold harmonics on a dragon mesh. The per vertex values v of these harmonic base vectors are mapped to colors via $r = \min(2 \cdot \max(\tilde{v}, 0.1), 0.8)$, $g = \min(r, b)$ and $b = \min(2 \cdot \max(1 - \tilde{v}, 0.1), 0.8)$, where $\tilde{v} = (v - min)/(max - min)$.

Applications of the Mesh Laplacian

The cotangent Laplacian plays an important role in many geometry processing applications. It is used in the context of smoothing, surface parametrization, for signal processing on manifolds, deformation and other areas. In this assignment you will explore some of the properties of the Laplacian. In the exercises 1-3 you will implement discrete Laplacians and use the mean curvature property of the Laplacian to smooth meshes, to enhace details and to compute minimal surfaces. In the fourth exercise you will generalize the Fourier transform to discrete surfaces, by mapping per vertex values to the base given by the eigenvectors of the Laplacian. On manifolds, this transform runs under the term manifold harmonics transform; the eigenvectors or eigenfunctions of the Laplacian operator are called manifold harmonics.

Solving Linear Equations

In the basecode_assignment4.zip package two solvers are provided, which you can use to solve the linear equations arising in this assignment. They both implement the same interface, so you can switch between them transparently. The `SciPySolver` calls the python script used

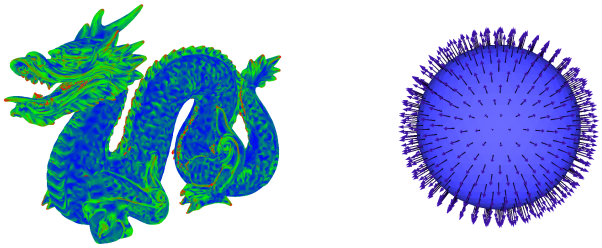

Figure 2: Left: Magnitude of the curvature normals computed with the cotangent Laplacian, colormapped with $r = (v >= 1\ ?\ v - 1 : 0)$, $g = (|v - 1| < 1\ ?\ 1 - |1 - v| : 0)$ and $b = (v < 1\ ?\ 1 - v : 0)$, where $v = log(1 + curvature/10)$. Right: curvature normals are rendered on the `sphere.obj` mesh.

in the last assignment; the `JMTSolver` uses a stabilized biconjugate gradient algorithm from the Java Matrix Toolkit and takes the right hand side as an initial guess. While this is much faster and works for the linear equations from this assignment, the solver sometimes fails to converge.

1 Laplacian Matrices

1. The two most widely used discretizations of the Laplace operator are the uniform (or graph) Laplacian and the cotangent Laplacian. Both discretizations of the Laplacian where introduced in the mesh processing book, in the differential geometry chapter. You can find the definition of the uniform Laplacian on page 44 and the definition of the cotangent Laplacian on page 46.

   (a) Implement the uniform Laplacian as a sparse $|vertex| \times |vertex|$ matrix. For each vertex v it will have a row of the type $(\cdots, -\frac{1}{|\mathcal{N}_v|}, \cdots, 1, \cdots, -\frac{1}{|\mathcal{N}|}, \cdots)$, where $|\mathcal{N}|$ is the valence of v.

   (b) Implement the cotangent Laplacian as a sparse $|vertex| \times |vertex|$ matrix. The cotangent Laplacian is described in formula 3.11 on page 46. Use mixed Voronoi areas as $\mathcal{A}$ in the normalization term (see Assignment 1 for details) . Note that you already computed all the weights you need in the assignment 1, exercise 4.

   Hint: For numerical stability it is worth to clamp cotangents when angles are close to zero, clamping the absolute values to $1e2$ *works well in practice.*

2. Check that your Laplacians have the following properties:

   (a) For both Laplacian matrices, each row has to sum to zero. If the per-row normalization terms $\frac{1}{|\mathcal{N}|}$ and $\frac{1}{\mathcal{A}}$ are dropped, they should additionally be symmetric.

   (b) Check the mean curvature property of the cotangent Laplacian L^{cotan}. Remember: the per-vertex mean curvature normals are computed by applying the discrete

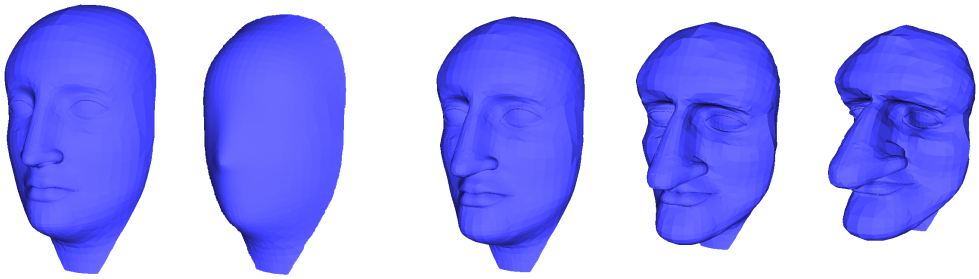

Figure 3: Unsharp masking is used to emphasis the details of the `head.obj` mesh. From left to right: The original mesh, the smoothed mesh (cotangent Laplacian, $\lambda = 0.1$), and the unsharp masked meshes with $s = 1.5, 2.5$ ans 3.5 respectively.

Laplacian defined for the mesh S to the coordinates of the mesh S

$$H = \frac{1}{2} \begin{pmatrix} L_S^{cotan} p.x \\ L_S^{cotan} p.y \\ L_S^{cotan} p.z \end{pmatrix}$$

and the mean curvature is given by the length of the mean curvature normal, $||H||$. Test the mean curvature property on the `sphere.obj` mesh, which represents a sphere of radius 2. The mean curvature of the sphere should be $\frac{1}{r}$, where r is the radius of the sphere. Also, check that the mean curvature normals are normal to the sphere.

(c) Visualize the mean curvature and the mean curvature normals computed using the cotangent Laplacian matrix for arbitrary meshes (see Figure 2). The results should coincide with the results computed in the exercise 4 of the assignment 1. Also, demonstrate how the *normed* curvature normals look, when the cotangent Laplacian is replaced by the uniform Laplacian; the `uglysphere.obj` mesh is a good test object.

3 Points

2 Smoothing and Unsharp Masking

Once the Laplacian matrices are defined, a plenitude of geometry processing algorithms can be implemented easily. One classic application is smoothing.

1. Implement the implicit smoothing scheme discussed in the lecture, i.e. solve

$$(I - \lambda L)p' = p$$

for x, y and z coordinates respectively. Here $\lambda = \mu dt$, in the lectures notation.

To prevent shrinkage, adapt your algorithm to preserve the volume of the mesh: after the smoothing rescale the mesh to have the same volume as before. The volume of an

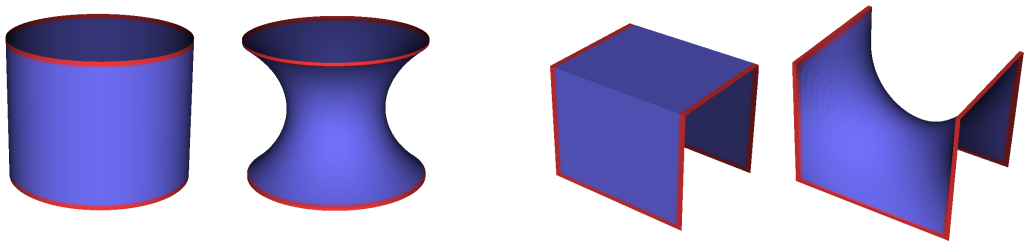

Figure 4: Pairs of surfaces and minimal surfaces that share the same boundary and topology. The two surfaces depicted here can be generted using the classes `Cylinder.java` and `Bock.java`

oriented mesh can be computed by

$$\sum_{triangles\{p1,p2,p3\}} \langle p1, p2 \times p3 \rangle / 6$$

Note that a single smoothing step is enough. Compare the quality of your smoothing algorithm using both the cotangent and the uniform Laplacian on the uglySphere.obj and some other meshes. For the cotangent Laplacian λ values in the range $(0.0001, 0.1)$ produce good results, for the uniform Laplacian λ values between 0.1 and 100 work well.

2. Implement the unsharp masking algorithm discussed in the lecture, see also Figure 3 for sample results. You are still very welcome to use the 3D scanner to produce a head mesh of your own, else a `head.obj` mesh is provided in the basecode_assignment4.zip package.

2 Points

3 Minimal Surfaces

1. Minimal surfaces are surfaces whose area is minimal, given some boundary constraint. They arise in architecture and describe the shape of soap bubbles (see `http://en.wikipedia.org/wiki/Minimal_surface`).

   It can be shown that minimal surfaces have zero mean curvature. On an intuitive level this is because the mean curvature normal field is something similar to a gradient for the surface area functional: the mean curvature normals point in the direction such that changing the positions in this direction, with the speed prescribed by the length of the normals, optimally minimizes the area. And therefore a surface can only be minimal if it has zero mean curvature. See `http://rkneufeld.wordpress.com/2010/10/27/minimal-surfaces-and-the-area-functional-2/` for a derivation.

   Your task is to compute minimal surfaces. Starting with some arbitrary mesh, fix the boundary and solve for the positions that have zero curvature

$$\begin{aligned} \Delta_{cotan}\xi = 0 \quad & \text{for none boundary vertices} \\ \xi = pos \quad & \text{on the boundary,} \end{aligned}$$

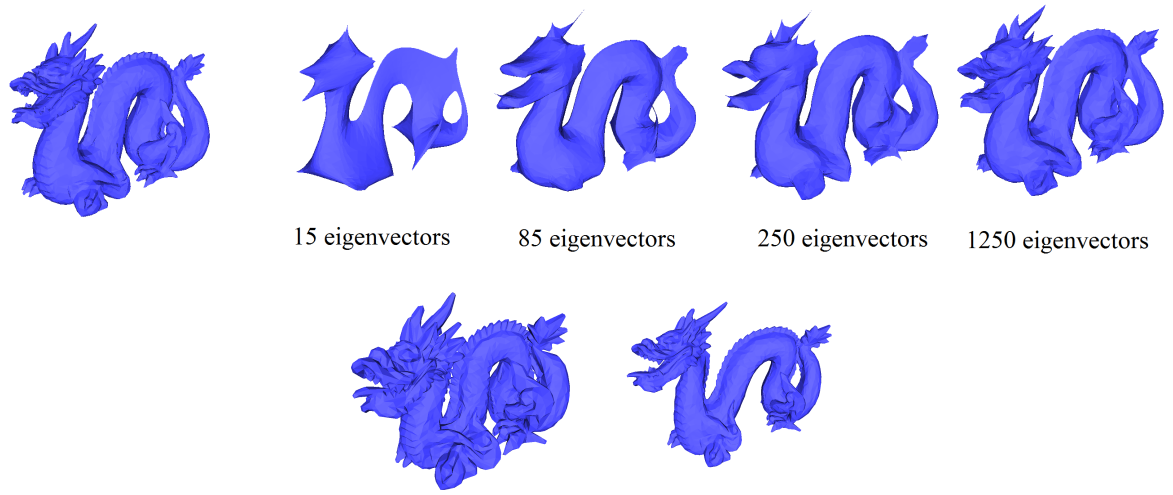

Figure 5: Top: Smoothing the dragon mesh by projecting it on a varying number of manifold harmonics base vectors. Bottom: Spectral filtering, high frequencies are enhanced (left), the frequency band between 5 and 10 is damped (right).

this amounts to setting thenrows of the Laplacian that correspond to boundary vertices to the identity and solve each coordinate for the appropriate right-hand side. After each solve, update the Laplacian and solve again for a surface with zero curvature. Iterate until the surface area is decreasing less than one percent in an one iteration step.

2 Points

4 Spectral Mesh Processing

In this exercise we explore some of the applications of spectral mesh processing. Spectral mesh processing has an interesting theoretical background because of its connection to Fourier transforms via eigenfunctions of the Laplacian. The main challenge in practice is that the computation of eigenvectors and eigenvalue pairs is quite complex, so you have to resort to specialized numerical methods to robustly find them.

In this exercise, to find the eigenvalue decomposition of a matrix, you can use the class `SCIPYEVD.java`- which again calls a SciPy script. To robustly find the eigenvalues, the script computes the full set of eigenvalues and eigenvectors, without making use of the sparseness of the matrices treated. This means that you will only be able to work with small meshes with 10'000 vertices tops. Such meshes would be `bunny.obj`, `teapot.obj`, `dragon_5k.obj` (provided in this assignments basecode package) and `sphere.obj`.

You can dazzle your teaching assistant by implementing the paper `http://www.cs.jhu.edu/~misha/ReadingSeminar/Papers/Vallet08.pdf` and demonstrate spectral processing on large meshes.

1. In order to make sure that all eigenvalues and eigenvectors exist, you need to implement a slightly different, symmetric cotangent Laplacian. Instead of scaling a row i by $\frac{1}{\mathcal{A}_i}$, where $\mathcal{A}_i$ is the mixed area around the vertex i, scale each entry $\Delta_i j$ with the weights

$\frac{1}{\sqrt{\mathcal{A}_i \mathcal{A}_j}}$ and set the diagonal entries such that each row sums to zero

$$\Delta_{ij} = -\frac{1}{\sqrt{\mathcal{A}_i \mathcal{A}_j}}(cotan(P) + cotan(Q))$$
$$\Delta_{ii} = -\sum_{j \neq i} \Delta_{ij}$$

Check that your matrix is symmetric and that every row sums to zero.

2. Compute and visualize the first 20 eigenvectors of the Laplacian on the `sphere.obj` mesh (see also Figure 1). These manifold harmonics run under the name *spherical harmonics*, which can for example be used to compress data stored on a sphere, but also describe electron configurations of atoms, see `http://en.wikipedia.org/wiki/Spherical_harmonics`. Note that spherical harmonics can be expressed analytically, so if you ever encounter them, use a closed formula to compute them, not some discrete Laplacian.

3. Spectral smoothing: use only the first k eigenvectors to smooth the mesh as described in the lecture. Using all eigenvectors should produce the original mesh.

4. The eigenvalues are related to the frequencies of the harmonic eigenfunctions. More precisely, the square root of the (absolute) eigenvalue denotes the frequency of the corresponding eigenvector. Similarly to filtering in the Fourier domain, you can filter in the spectral domain; after the projection onto the eigenvectors, scale each entry i with a weight f depending only on the frequency of the eigenvector e_i:

$$\begin{pmatrix} e_1 & \cdots & e_n \end{pmatrix} \begin{pmatrix} f(freq(e_1)) & & & \\ & f(freq(e_2)) & & \\ & & \ddots & \\ & & & f(freq(e_n)) \end{pmatrix} \begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix} (x, y, z) = (\tilde{x}, \tilde{y}, \tilde{z})$$

Experiment with at least three different filters that damp and enhance certain frequencies. Ideas: enhance or dampen high, low or medium frequencies, drop frequencies, combine sigmoids, combine any two filters using max or min to produce a more complex filter.

3 Points

5 Hand-In

Don't forget to:

1. Commit your code via the Ilias exercise page before the deadline.

2. Prepare your code demonstration and reserve a time slot for your demo via the Google Doc shared in the forum.