Answers to COMP0119 (Course Work 2)

Chadjiminas Chrysostomos **Student Id No.:** [redacted]

1 Questions attempted

I have attempted all the questions asked.

1.1 Question 1

Estimated discrete Gaussian and mean curvature, visualised and compared results.

1.2 Question 2

Constructed Laplace Beltrami and re-estimated mean curvature and compared results.

1.3 Question 3

Computed and showed mesh reconstructions using the k smallest eigenvectors.

1.4 Question 4

Implemented explicit Laplacian mesh smoothing. Presented problems with high λ and provided good smoothing results with good λ.

1.5 Question 5

Implemented implicit Laplacian mesh smoothing. Compared results with explicit smoothing.

1.6 Question 6

Evaluated Laplacian smoothing in the context of data smoothing by applying increasingly more noise to meshes and then trying to recreate the original meshes and reporting when the method fails and when it succeeds.

2 Discrete Curvature and Spectral Meshes

2.1 Mean and Gaussian curvature

There are two ways to measure the curvature of a surface, mean and Gaussian curvature. Both have different properties and uses.

Mean curvature is measured as:

$$H = \frac{\kappa_1 + \kappa_2}{2}$$

And Gaussian curvature as:

$$K = \kappa_1 \cdot \kappa_2$$

Because the meshes are discrete we can approximate those values.

The mean curvature can be approximate as:

$$H = \frac{1}{2}||\Delta_S \mathbf{x}||$$

Where Δ_S is a discreet Laplace operator.

The Gaussian curvature can be approximated as:

$$K = \frac{(2\pi - \sum_j \theta)}{A}$$

Where $2\pi - \sum_j \theta$ is the angle deficit at each vertex and A can be either barycentric or voronoi area of the faces in the neighborhood as discussed in the lectures. The areas used in this coursework are the barycentric areas.

We can see the results for Gaussian mean on some of the meshes in fig 1.

We can see the results for the mean curvature (for uniform and cotangent Laplace operator) in figures 2 and 3.

As expected we can see that the Gaussian curvature has high values where both k1 and k2 have high values and the same sign. In the bumps of the bumpy cube we can see this effect. In the base of the bumps we can see that we have a hyperbolic (or saddle) point where we have positive and negative curvature we gives us a negative value and therefore we can see really dark colour signifying a small value. On the flat surfaces of the cube we can see that we have a constant blue color which is probably for values close to zeros because the positive and the negative curvature are close to zero.

As for the mean approximation we can see that we have high values on the bumps of the cube where we have a high curvature in the bumps. I think that mean curvature

approximations have better visual results because we can see that where the meshes have a bumps (hot areas) and where they are flat (blue areas). The uniform mean curvature is not be a very good approximations of the continuous curvature because the uniform Laplace operator is based only on the connectivity of the mesh. We can see these artefacts in the bumpy cube in figure 2 where we can see that in places were supposed to be flat we have negative or positive curvature. This problem is solved by using the cotan discretisation of the Laplace operator.

2.2 The Laplace operator

The Laplace operator is a differential operator that can be applied on a multivariate differentiable function to get the divergence of the gradient. The Laplace operator is symbolised with Δ:

$$\Delta f(p) = \nabla \cdot \nabla f(p)$$

where ∇ is the gradient operator.

Mesh manifolds are such functions and we can use the Laplace operator to do many operations on them such as calculating the mean curvature, mesh reconstruction and smoothing. Meshes are not continuous and are comprised of a finite amount of vertices. Therefore, we can approximate the Laplace operator for a mesh with many discretisations.

The discrete Laplace operator is related to the mean curvature H with this operation.

$$\Delta \mathbf{x} = \begin{bmatrix} \frac{\partial f(\mathbf{f}(x)}{x_1} \\ \frac{\partial f(\mathbf{f}(x)}{x_2} \\ \frac{\partial f(\mathbf{f}(x)}{x_3} \\ \vdots \\ \frac{\partial f(\mathbf{f}(x)}{x_n} \end{bmatrix} = -2Hn$$

By calculating the Laplace operator we can extract the discretised mean curvature as:

$$H = \frac{1}{2}||\Delta_S \mathbf{x}||$$

Intuitively, $\Delta \mathbf{x}$ gives a measure of detail at each vertex in the mesh.

In the next section we present the uniform and Laplace Beltrami discretisations and the discretised mean curvature extracted from them visualised on the meshes.

2.3 Uniform Laplace discretisation of the Laplace operator

One of the discretisations is the uniform Laplace discretisation that can be calculated as:

$$\Delta_{\text{uni}} f(v_i) = \frac{1}{||N_1(v_i)||} \sum_{v_j \in N_1(v_i)} (f(v_j) - f(v_i))$$

The uniform operator depends only on the connectivity of the mesh and is a bad approximation when you have irregular triangulation. This effect is apparent when we see the mean curvature in the bumpy cube in figure 2.

2.4 Cotan discretisation of the Laplace operator. Laplace Beltrami

We construct the Laplace beltrami operator as:

$$\Delta_S f(v_i) = \frac{1}{2A(v_i)} \sum_{v_j \in N_1(v_i)} (cot\alpha_{ij} + cot\beta_{ij})(f(v_j) - f(v_i))$$

where $A_{(vi)}$ is the barycentric area of the faces that are adjacent to the vertex. That is

$$A(v_i) = \sum_j \frac{A_j}{3}$$

where A_j is the surface of each face.

With this discretisation we fix the problems that were presented with the uniform operator because the discretisation no longer depends on the connectivity only but also the areas of each face (or rather the barycentric area). We can see that the results are better by observing the mean curvature of the bumpy cube in figure 3 where the artefacts that were observed because of the irregular triangle connectivity are no longer there.

2.5 Mesh reconstruction

The Laplacian operator can also be used for mesh reconstruction in the context of spectral analysis. As previously described the operation $\Delta\mathbf{x}$ gives us a measure of detail for each vertex. To be more precise it gives us a vector that is opposite to the normal of the vertex scaled by the the mean curvature. :e can compute eigendecomposition on the Laplace operator and we can extract the k smallest eigenvectors (ranked by their corresponding eigenvalues) and reconstruct the mesh with the following operation from the slides:

$$\mathbf{x} := [x_1, \ldots, x_n] \qquad \mathbf{y} := [y_1, \ldots, y_n] \qquad \mathbf{z} := [z_1, \ldots, z_n]$$
$$\mathbf{x} \leftarrow \sum_{i=1}^{k} \left(\mathbf{x}^T \mathbf{e}_i\right) \mathbf{e}_i \qquad \mathbf{y} \leftarrow \sum_{i=1}^{k} \left(\mathbf{y}^T \mathbf{e}_i\right) \mathbf{e}_i \qquad \mathbf{z} \leftarrow \sum_{i=1}^{k} \left(\mathbf{z}^T \mathbf{e}_i\right) \mathbf{e}_i$$

Unfortunately the Laplacian operator is not symmetric so it's hard to do the eigendecomposition directly on it. In this coursework we used another method described in [1] and also in the slides where we use the diagonal matrix M and the symmetric matrix C (the decomposed Laplacian matrix) to calculate the eigendecomposition on a symmetric matrix and then extract the original eigenvectors from this result, for more information please see the paper.

In brief we construct the matrix A as:

$$A = M^{-\frac{1}{2}} C M^{-\frac{1}{2}}$$

We do eigendecomposition on A to get k eigenvectors and then map the given eigenvectors to the canonical basis:

$$\phi_i = M^{-\frac{1}{2}} \psi_i$$

where phi_i are the projected eigenvectors and psi_i are the eigenvectors from the eigendecomposition. Then we reconstruct the meshes with the operation described above using the k eigenvectors to get a reconstructed mesh.

The results for reconstruction can be seen in figure 4. We can see that 5-30 eigenvectors are not enough to get the more salient details of the mesh but they do capture the lowest frequencies. We can see that for k=30 we get to see the main cube being formed and some of the bumps starting to show up. In figure 5 we can see that with more eigenvectors we can see the finer details of the cube and we almost get the original mesh.

The results for eigen reconstruction for the cow mesh can be seen in figure 6

3 Laplacian Mesh smoothing

3.1 Explicit mesh smoothing

For explicit smoothing we apply the following operation to the mesh vertices each iteration:

$$P^{(t+1)} = (I + \lambda L) P^{(t)}$$

Applying explicit smoothing we get the results shown in 8. The choice on λ and iterations was made in a way to produce good results while the algorithm runs stably but also making sure not too many iterations are needed. With big λ we observed that we got unstable results that can be seen in figure 7

The best results were taken when we had really small λ and a large amount of iterations. Unfortunately, this would take too much time and therefore we had to make a compromise between λ and iterations. For the bunny mesh shown in 8 a good amount for λ was found out to be 0.005.

3.2 Implicit mesh smoothing

For implicit mesh smoothing we solve the linear system

$$(I - \lambda L)P^{(t+1)} = P^{(t)}$$

each iteration. Because $L = M^{-1}L_w$ is not symmetric we can multiply M in both sides and we can turn this problem to:

$$(M - \lambda L_w)P^{(t+1)} = MP^{(t)}$$

And then use a sparse solver to get the vertices at the next iteration. By applying this operation many times we get a smoother result each time. Results for implicit mesh smoothing can be seen in figures 9. Additional results for smoothing can be seen in figures 11121314 where we smooth noisy meshes.

Comparing the implicit with the explicit smoothing results we got, the results are pretty similar. The explicit smoothing fails when λ is high as shown in 7. In implicit smoothing we also realised that, although integration is unconditionally stable, λ were required to be really small because it made meshes too smooth with less iterations than with the explicit smoothing.

3.3 Denoising of the mesh using Laplacian smoothing

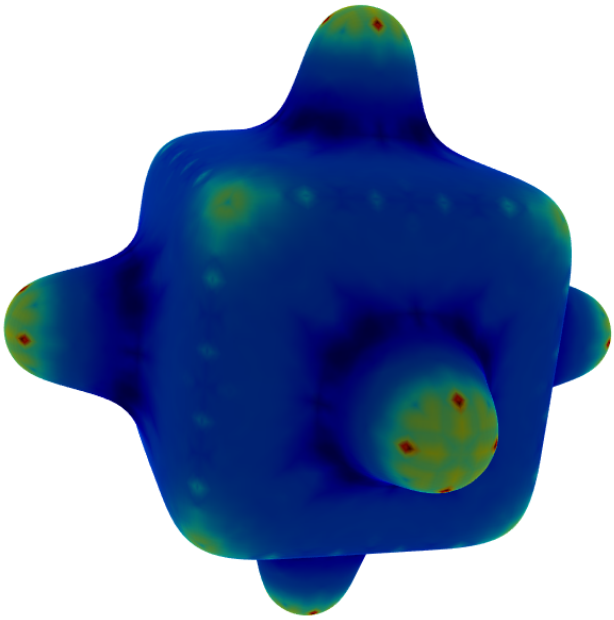
Laplacian smoothing can also be used to denoise a mesh. To experiment denoising a mesh we add random displacement to the vertices of the mesh, smooth out the noisy mesh in an attempt to recover the original mesh by trying different lamda values and iterations until we get a good result and then we evaluate the result. We repeated this process with

increasing amount of noise. The results can be seen in figure10. We can observe that for low amounts of noise we can recover the original mesh without losing important features such as the shape of the legs of the cow. When the noise becomes too much, we can see that we lose important details of the mesh before we recover the details for some others. For example, in the cow mesh demonstrated in the figure we can see that when the noise original mesh is big, when we smooth the mesh we loose important details in the feet of the cow where they lose their shape while the face remains noisy and jaggy.

For additional results of denoising and smoothing on noisy example meshes please see figures 11 13 14.

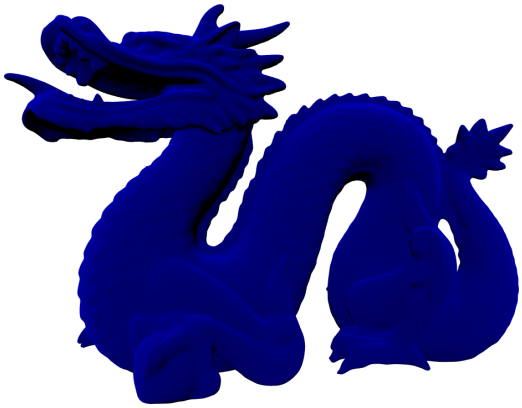
References

[1] B. Vallet and B. Lévy. "Spectral Geometry Processing with Manifold Harmonics". en. In: *Computer Graphics Forum* 27.2 (Apr. 2008), pp. 251–260. ISSN: 0167-7055, 1467-8659. DOI: 10.1111/j.1467-8659.2008.01122.x. URL: http://doi.wiley.com/10.1111/j.1467-8659.2008.01122.x (visited on 03/25/2020).



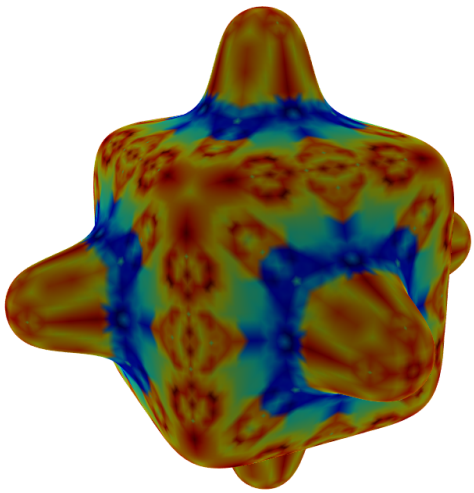
(a) Gaussian curvature bumpy cube.

(b) Gaussian curvature cow.

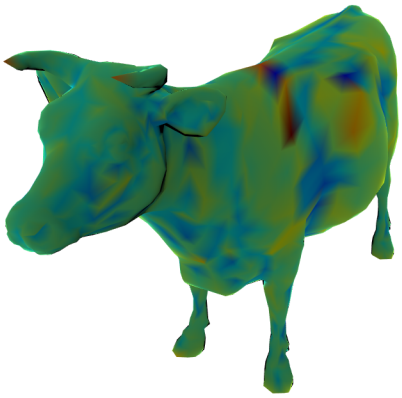


(c) Gaussian curvature dragon.

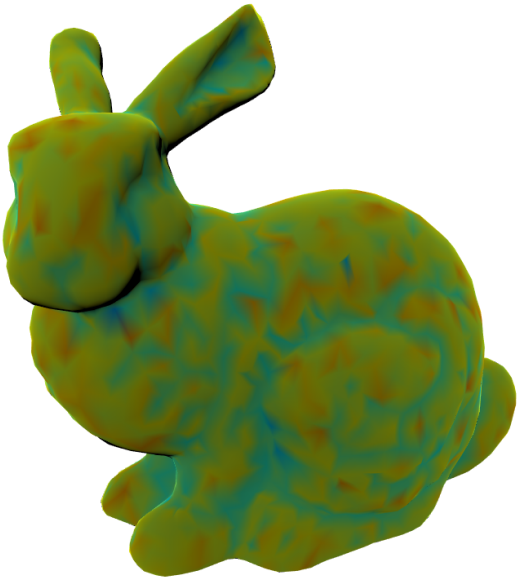
Figure 1



(a) Mean curvature bumpy cube, uniform Laplace.

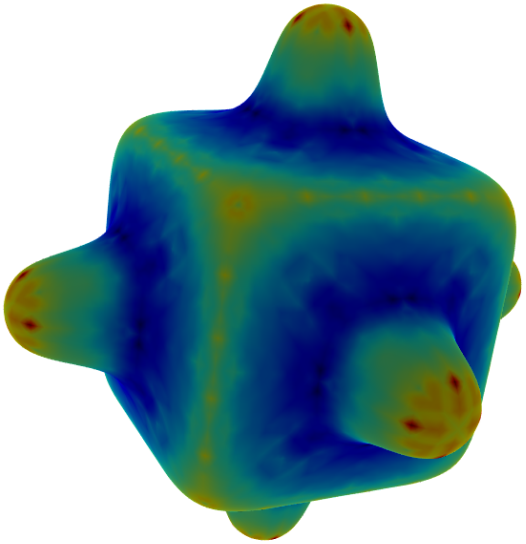


(b) Mean curvature cow, uniform Laplace.



(c) Mean curvature bunny, uniform Laplace.

Figure 2: Uniform Laplace operator, mean curvature

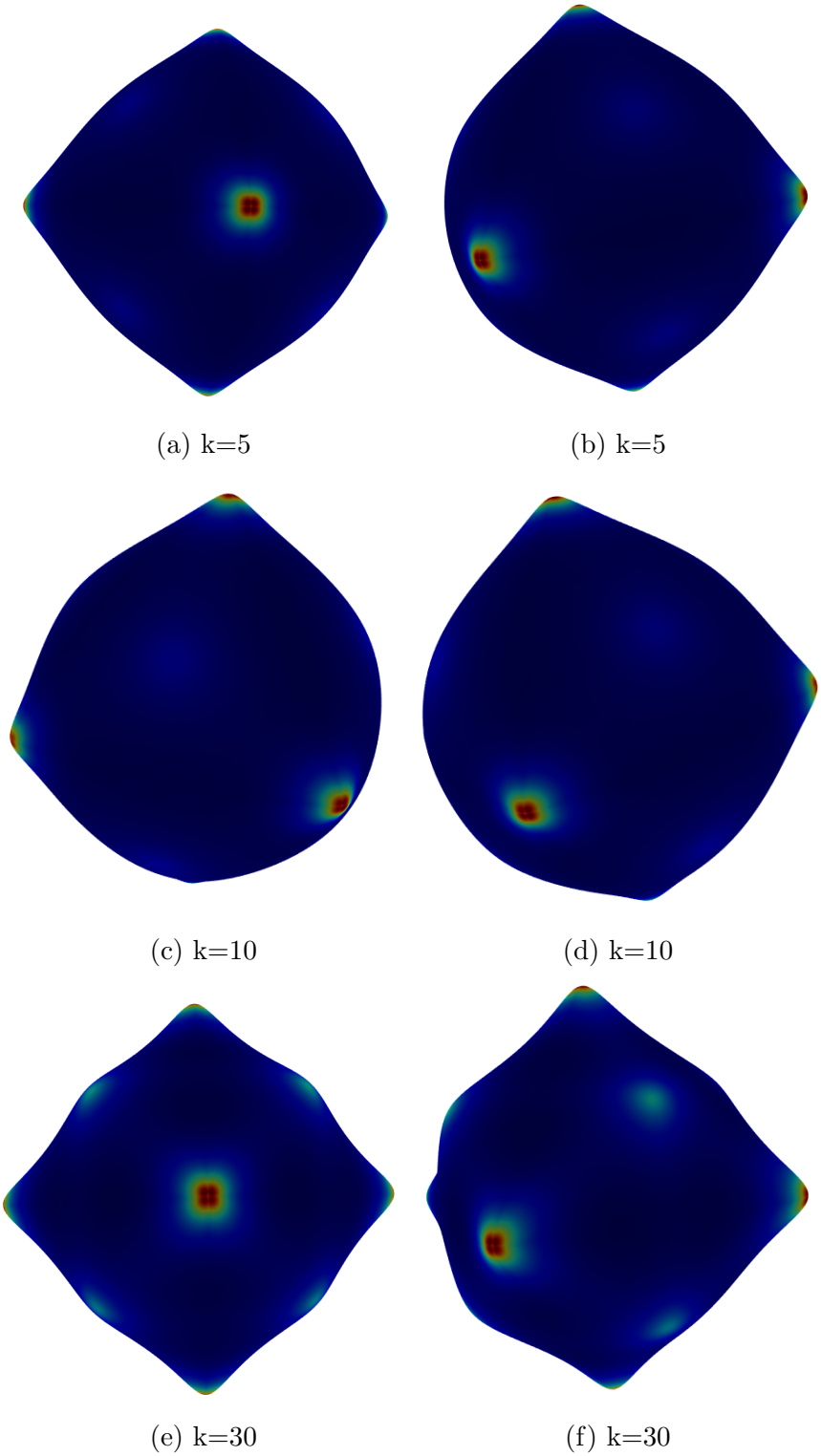


(a) Mean curvature bumpy cube, cotan Laplace.

(b) Mean curvature bumpy cube, cotan Laplace.

(c) Mean curvature bunny, cotan Laplace.

Figure 3: Mean curvature with cotan Laplace operator results.



(a) k=5 (b) k=5

(c) k=10 (d) k=10

(e) k=30 (f) k=30

Figure 4: Bumpy cube reconstructions for k 5, 10 and 30

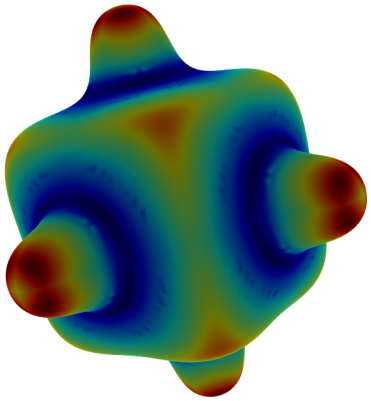
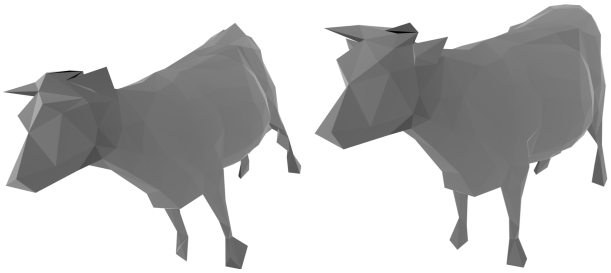


Figure 5: Mesh reconstruction with k=130

(a) k=5 (b) k=10 (c) k=30



(d) k=250 (e) k=300

Figure 6: Cow reconstruction with increasing k.

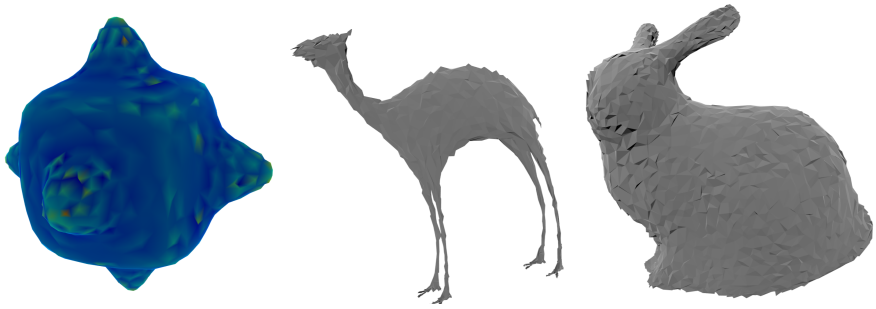


Figure 7: Bad explicit smoothing with high λ values (1.5) and 10 iterations.

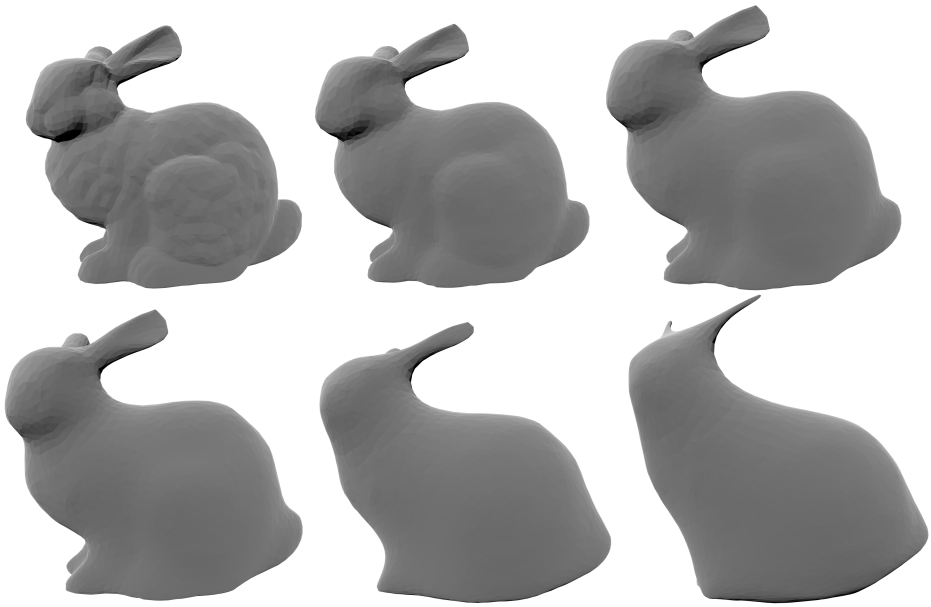


Figure 8: Explicit smoothing. From least smooth to most smooth mesh from left to right top to bottom. We can increase smoothness by increasing iterations or increasing λ. High λ values produce bad results and really low λ, although they produce better results and provide finer smoothing control can take much more iterations to produce similar results and can take a large amount of time. λ used for these results are 0.05 and 0.005 and, depending on λ, we use from 100 to 1500 iterations.

Figure 9: Implicit smoothing. From least smooth to most smooth mesh from left to right top to bottom. As with implicit smoothing we can increase smoothness by increasing iterations or increasing λ. λ used are $5x10^{-8}$ for 50 to 5000 iterations.

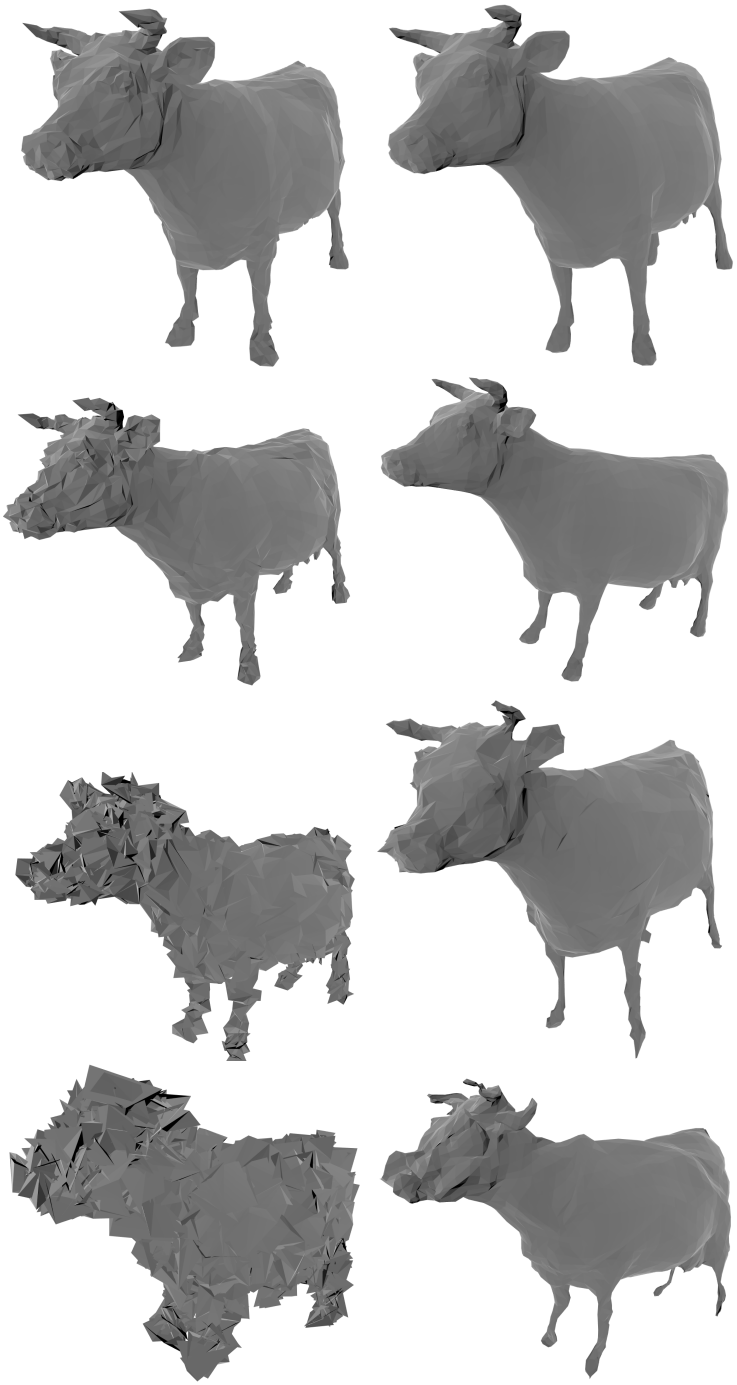
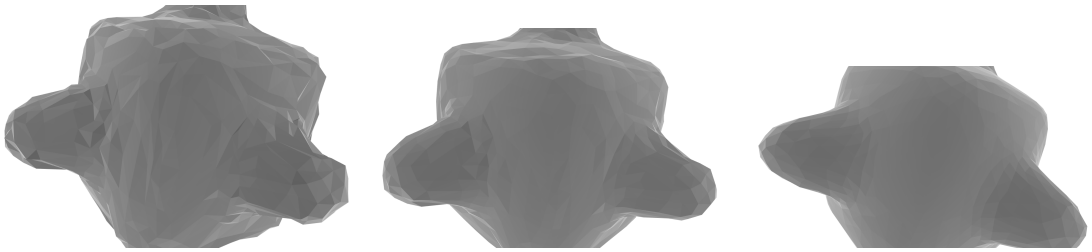
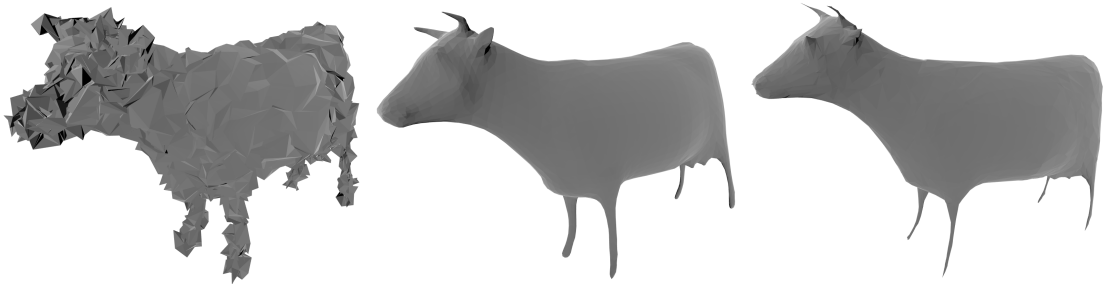


Figure 10: On the left is the noisy mesh and on the right is the corresponding denoised mesh. We can see that as noise denoising the mesh harder. We can see that during the denoising using smoothing important details are lost when the noise is too much.



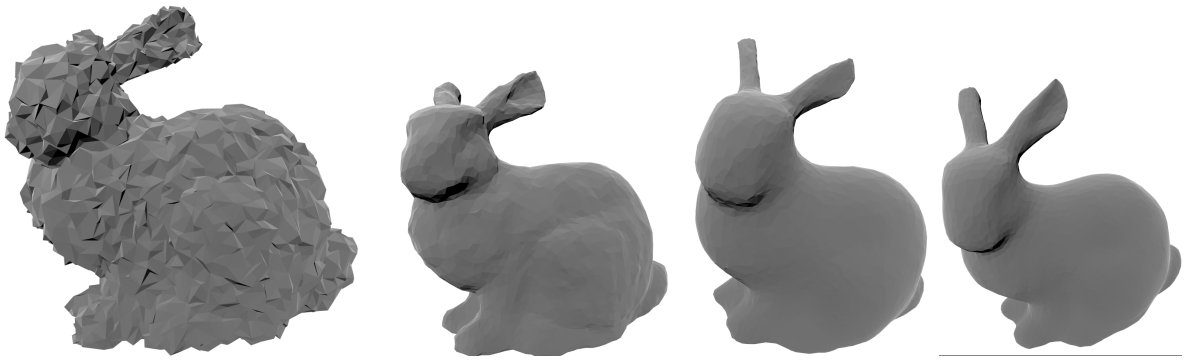
(a) Noisy small bumpy cube. (b) Noisy bumpy cube smoothed 1000 iterations (c) Noisy bumpy cube smoothed 30000 iterations.

Figure 11: Small noisy bumpy cube smoothing



(a) Noisy cow. (b) Explicit smoothing. (c) Implicit smoothing

Figure 12: Smoothing of noisy cow.



(a) Noisy bunny (b) Denoised bunny (c) Noisy bunny smoothing implicit. (d) Noisy explicit smoothing.

Figure 13: Smoothing of noisy bunny.

(a) Noisy camel. (b) Denoised camel (c) implicit smoothing. (d) explicit smoothing

Figure 14: Smoothing of noisy camel.