COMPM080 - Report

LaPlace Beltrami operator

Fen Jin

March 2019

1 Introduction

The goal of this coursework is to build the LaPlace Beltrami (LPB or Δ) operator and apply it to calculate the mean curvature, do spectral analysis of the meshes and mesh smoothing.

In the first part, we will cover the uniform and cotan version of the LPB matrix, and the spectral analysis. While in the second part, we will cover LaPlacian mesh smoothing, both explicit and implicit version.

To demo, use the menu window (shown in Fig.1) to try the different modes (mean curvature from different version of LBP matrices, mesh smoothing and spectral analysis). We can choose which model to use (mode 0 -Resetting), and choose the values for how many eigenvalues to use in spectral analysis, the value for lambda in mesh smoothing and the amount of Gaussian noise to add to the mesh.

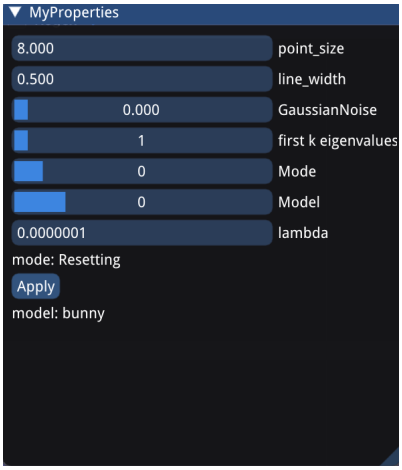


Figure 1: Menu window

In this report, we will use V to indicate the vertices, F to indicate the faces.

2 Part 1

2.1 Task 1

The first task is to build the uniform LPB operator; this is a symmetric sparse matrix, in which every row contains $|N_1(v_i)|+1$ non zero elements, where $|N_1(v_i)|$ is the number of one-ring neighbors

of vertex i. Since the mesh are composed by triangles, the average value of $|N_1(v_i)|$ will be 6 (valence).

The formulation we want to follow is:

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

where v indicates a vertex and f is a generic function (colors, normals, etc.) applied on the vertices, in our case we will use the vertices position. So, the resulting formulation will be:

$$\Delta_{uni} \mathbf{x}_i = \frac{1}{|N_1(v_i)|} \sum_{v_j \in N_1(v_i)} (\mathbf{x}_j - \mathbf{x}_i) = -2H\mathbf{n}$$

where $\mathbf{x}$ is the vertices position vector ($|V| \times 3$), $\mathbf{n}$ contains the normal vectors for each vertex ($|V| \times 3$) and H is the mean curvature.

To find the exact number of neighbors $|N_1(v_i)|$ for each vertex i, we loop for each vertex, we identify the connected faces, which contain the indices of its vertices. The number of faces connected to the current vertex corresponds also to the number of its adjacent vertices. In L_{ij}, for $j = 1 \dots N, j \neq i$, we put the value $-\frac{1}{|N_1(v_i)|}$, while in L_{ii} we put 1, which corresponds to $-\sum_j L_{ij}, j \neq i$, because the sum of each row needs to be zero. Basically, uniform LPB considers all the triangles to be very similar, i.e. the mesh to be quite regular. Then the LPB operator is the product of the inverse of the diagonal matrix M of areas A_i (explained later) and the L matrix.

$$M = \begin{pmatrix} A_1 & 0 & \dots & 0 \\ 0 & A_2 & & \vdots \\ \vdots & & \ddots & \\ 0 & \dots & & A_{|V|} \end{pmatrix}$$

So to sum up for the uniform LPB:

$$L_{ij} = -\frac{1}{|N_1(v_i)|}$$
$$L_{ii} = -\sum_j L_{ij} = 1, j \neq i$$
$$\Delta = M^{-1} L$$

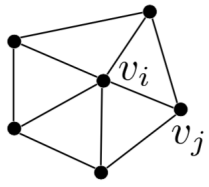


Figure 2: Example of a vertex i and its one ring neighbors.

The mean curvature H is then calculated using:

$$\Delta \mathbf{x} = -2H\mathbf{n} \tag{1}$$

We can visualize H by taking its magnitude:

$$|H| = \frac{|\Delta \mathbf{x}|}{2}$$

(a) Bunny (b) Camel (c) Cow

Figure 3: Magnitude of mean curvature calculated using uniform LPB.

Fig. 3 shows the mean curvature on some models.

To calculate the discrete Gaussian curvature, we need to calculate the angle deficit for each vertex:

$$K = \frac{2\pi - \sum_{j \in N_1(v_i)} \theta_j}{A_i}$$

where A_i is the area normalization factor. The area used in this implementation is the barycentric area, which corresponds to 1/3 of the area covered by all the one-ring neighbor faces of the considered vertex as seen in Fig.4. Other types of area per vertex exists and they lead to different normalization, but overall the result should not look qualitatively too different.

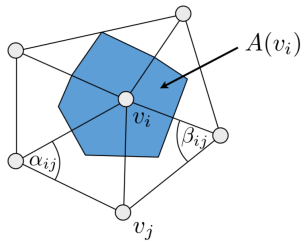


Figure 4: One ring neighbors of vertex i. The colored part is the area corresponding to v_i.

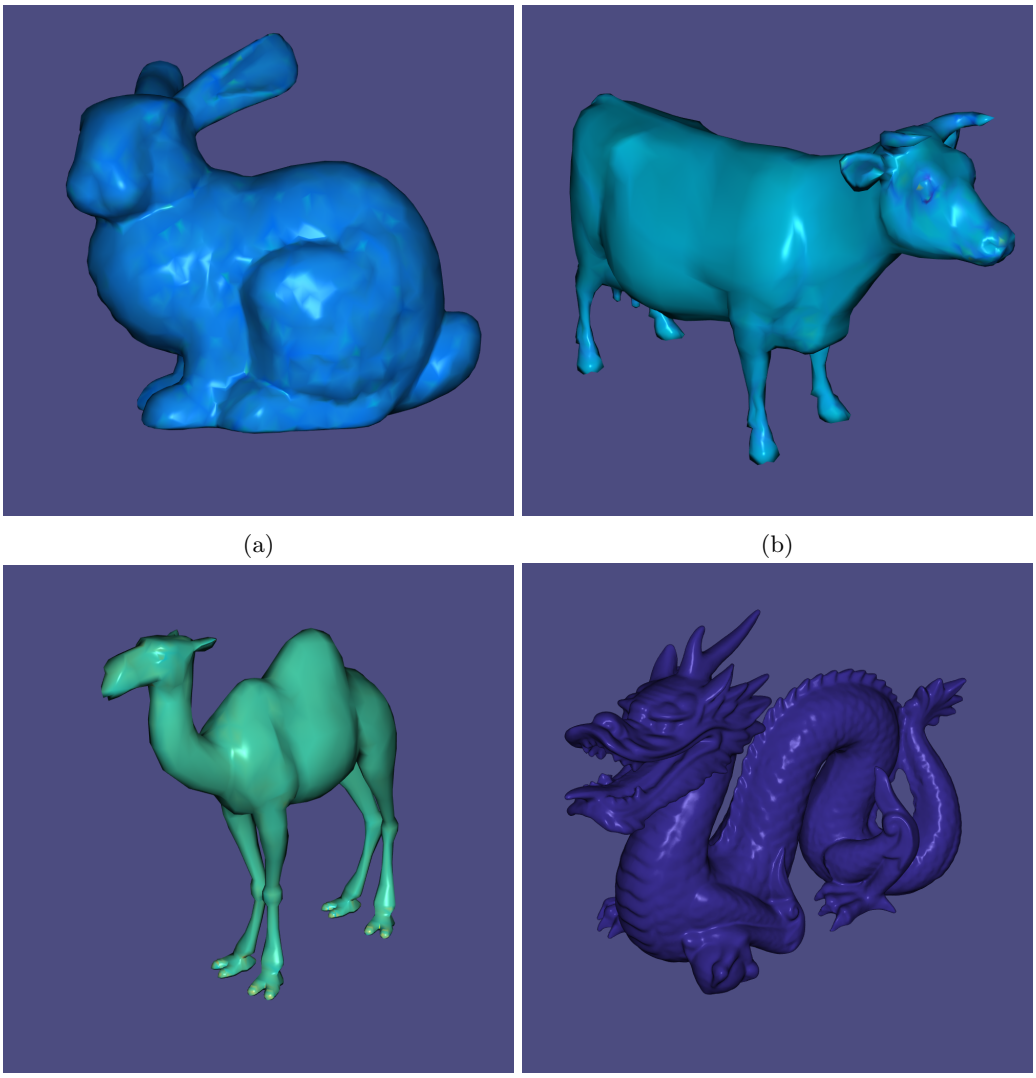
In Fig.5 we show some results of the Gaussian curvature for the models given. The results look correct, especially for Fig. 5a and 5c, we can notice that parts like the eye and legs of the cow have darker colors, while most of the body is plain and the color is mostly uniform.

So, I would say that this discretization approximates fairly good the continuous curvature.

2.2 Task 2

In this task, we have to compute the cotan version of the LPB operator. While the uniform LPB takes each neighbor of a vertex as equal to another neighbor of the same vertex ($\Delta_{ij} = \Delta_{ik} = -\frac{1}{|N_1(v_i)|}$ with $j, k \in N_1(v_i)$), the cotan LPB will take into account how the triangles in the meshes are shaped. The weight between v_i and v_j will be the sum of the cotan[1] of angles α_{ij} and β_{ij}, as

[1] $cotan(\theta) = (\frac{\cos\theta}{\sin\theta})$



(a) (b) (c) (d)

Figure 5: Discrete Gaussian curvature.

shown in Fig.4. If the mesh is irregular, which is most of the cases, the cotan LPB will consider better this irregularity.

$$c_{ij} = \frac{1}{2}(cotan(\alpha_{ij}) + cotan(\beta_{ij}))$$
$$c_{ii} = -\sum_{j \in N_1(v_i)} c_{ij}$$
$$\Delta_{cotan} = M^{-1}C$$

Then, the mean curvature H is calculated as before:

$$\Delta_{cotan}\mathbf{x} = -2H\mathbf{x}$$

Fig. 6 show results of mean curvature using cotan LPB.

(a) Bunny (b) Camel (c) Cow

Figure 6: Magnitude of mean curvature calculated using cotan LPB.

The results are better than the uniform LPB. We can notice that it is more stable in uniform areas, while in areas where the local curvature is higher, the mean curvature is estimated correctly, e.g. the horns of the cow.

2.3 Task 3

This task aims to do the spectral analysis. Given the Δ_{cotan} found in the previous task, we want to find the k smallest eigenvalues and the corresponding k eigenvectors. The idea is to decompose the mesh in its principal components space. As in Fourier domain, we can use different frequencies to reconstruct approximately the original signal, here we want to use only some eigenvectors to recompose approximately the original mesh. The smallest eigenvalues will build the low frequencies of the mesh, and every time we consider more following eigenvalues, it would be as adding higher frequencies. In the context of meshes, the frequencies can be interpreted as details (lower frequencies can be seen as coarser details). If k = n, then the reconstruction will be complete, the resulting mesh would be exactely the same as the original. The eigen decomposition we want to solve is:

$$\begin{gathered}\Delta\phi_i = \lambda_i\phi_i \\ M^{-1}C\phi_i = \lambda_i\phi_i \\ M^{-\frac{1}{2}}CM^{-\frac{1}{2}}M^{\frac{1}{2}}\phi_i = \lambda_i M^{\frac{1}{2}}\phi_i \\ \tilde{\Delta}y_i = \lambda_i y_i\end{gathered}$$

where

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

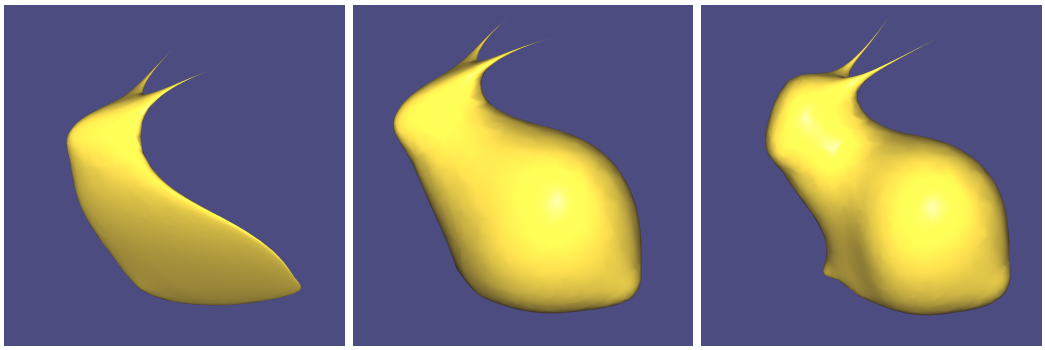
Once we have the ϕ_i, we can do the reconstruction by doing a weighted sum:

$$\tilde{\mathbf{x}} = \mathbf{x}^T M\phi = \sum_{i=1}^{k} \mathbf{x}_i M_{i,i}\phi_i$$

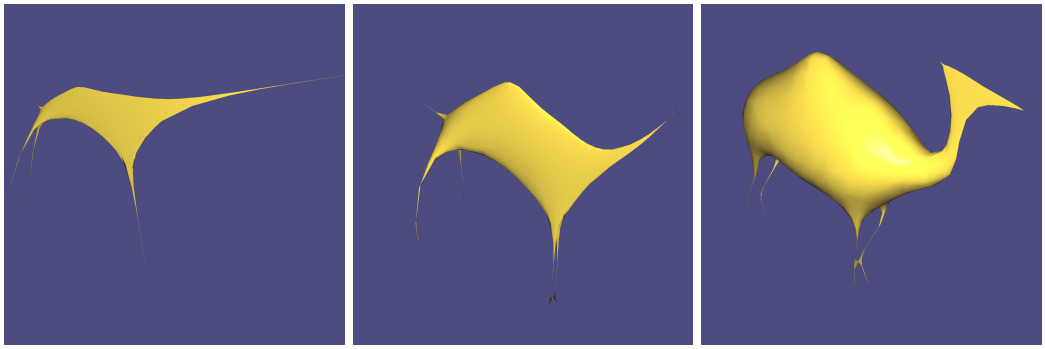
Fig. 7 shows some reconstruction, using the spectral analysis, for different numbers of eigenvalues.

2.4 Task 4

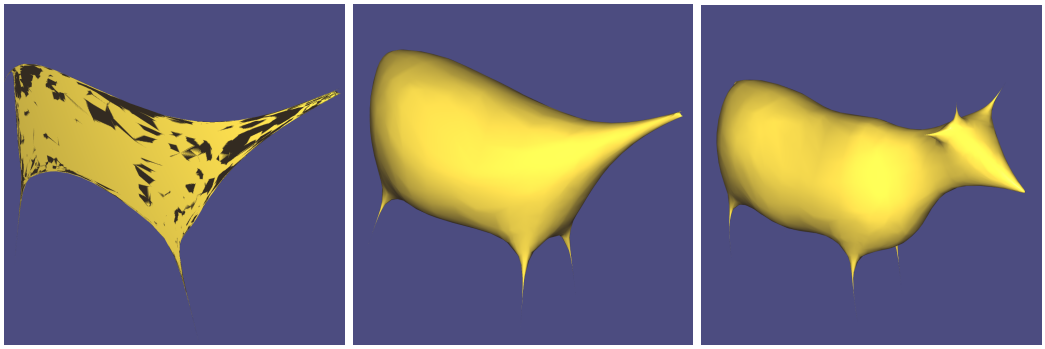
Demo Fig.8 shows the cow mesh reconstructed with the 10 smallest eigenvalues.



(a) Bunny 5 k. (b) Bunny 10 k. (c) Bunny 30 k.



(d) Camel 5 k. (e) Camel 10 k. (f) Camel 30 k.



(g) Cow 5 k. (h) Cow 10 k. (i) Cow 30 k.

Figure 7: Spectral analysis reconstruction using different number of eigenvalues. The eigenvalues used are the k smallest.

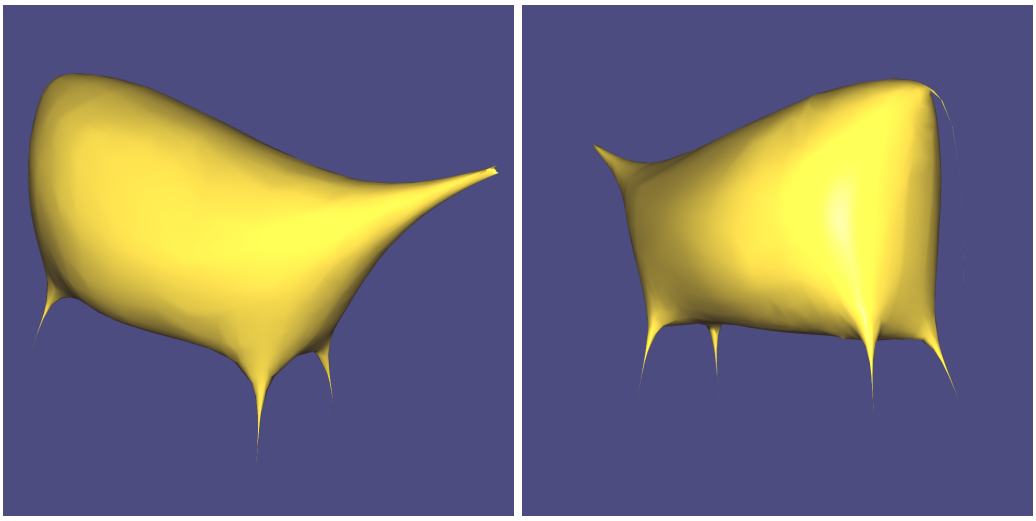
3 Part 2

3.1 Task 5

In this task we are doing mesh smoothing using the explicit form of LaPlacian smoothing. The method requires to iteratively move each vertex to in the direction opposite to the normals, by the magnitude given the mean curvature. It's exactly applying Eq.1, repeatedly to the vertices.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda \Delta \mathbf{x}_k \tag{2}$$

λ is a step factor, changing it will result in different outputs. The value of λ is quite important



(a) Front view. (b) Rear view.

Figure 8: Spectral analysis of the cow, using the first 10 eigenvalues.

because it scales how much we want to move, if it is too large then we would overstep and the smoothing becomes unstable, which means that some vertices are going to "explode" (see Fig.10j, 10k and 10l). A good value of λ seems to be 10^{-7}, for example in Fig.10g even after 200 iterations the mesh remains stable and it's correctly smoothed. Although, good values of λ seems to depend on the mesh itself, for example in Fig.10h and 10f even with higher values of λ the smoothing is not unstable.

Fig.9 and 10 shows some smoothing using this explicit form.

3.2 Task 6

In this task we will implement the implicit form of the LaPlacian smoothing. This is more stable and it allows to use higher values of λ.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda \Delta \mathbf{x}_{k+1} \tag{3}$$

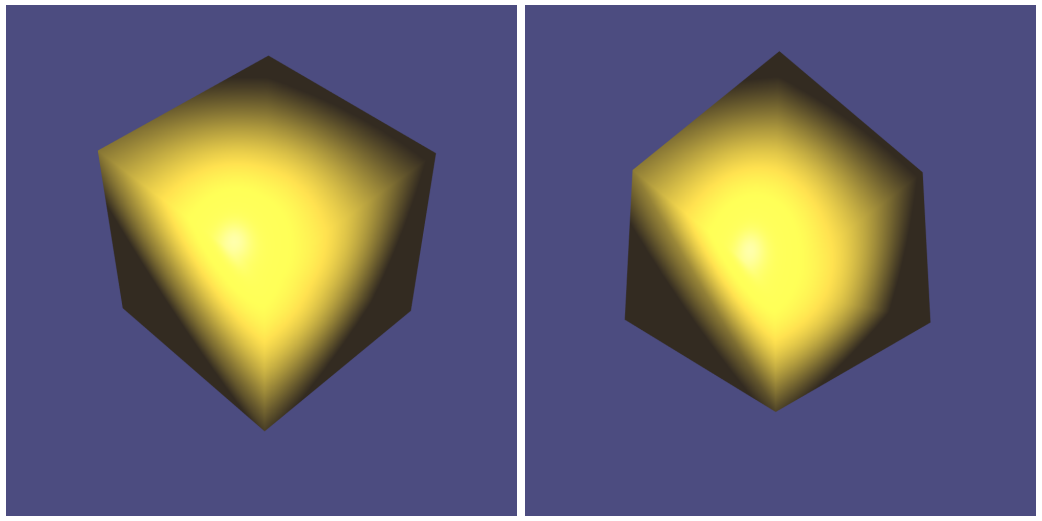
The difference is that instead of computing the new positions based only on the current step, the implicit form takes into account, at step k, also the position just calculated in this same step for the later vertices. From Eq.3, we can derive that:

$$(M - \lambda C)\mathbf{x}_{k+1} = M\mathbf{x}_k$$

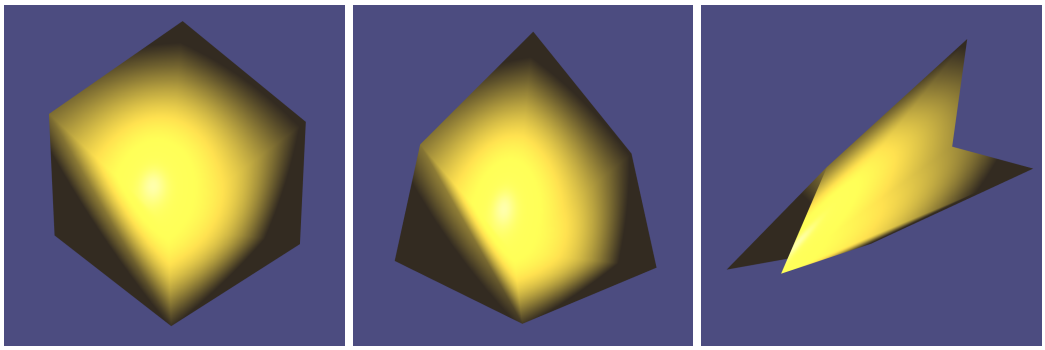
Fig.11 shows results using the implicit version of the LPB smoothing. We can notice that for the same value of λ and the same number of iterations, the implicit form is more stable. When we smooth a lot, for example Fig.11g and 11h, we see that the details are canceled and the results are similar to the reconstruction using the smallest eigenvalues.

3.3 Task 7

In this section we will evaluate the mesh smoothing effect when some noise is added to the meshes. Adding noise will make the meshes more spiky. The type of noise we will add is Gaussian noise,



(a) Original cube. (b) After 100 iterations.



(c) After 200 iterations. (d) After 300 iterations. (e) After 400 iterations.

Figure 9: Cube smoothing with $\lambda = 10^{-7}$. Here are shown progresses through the iterations.

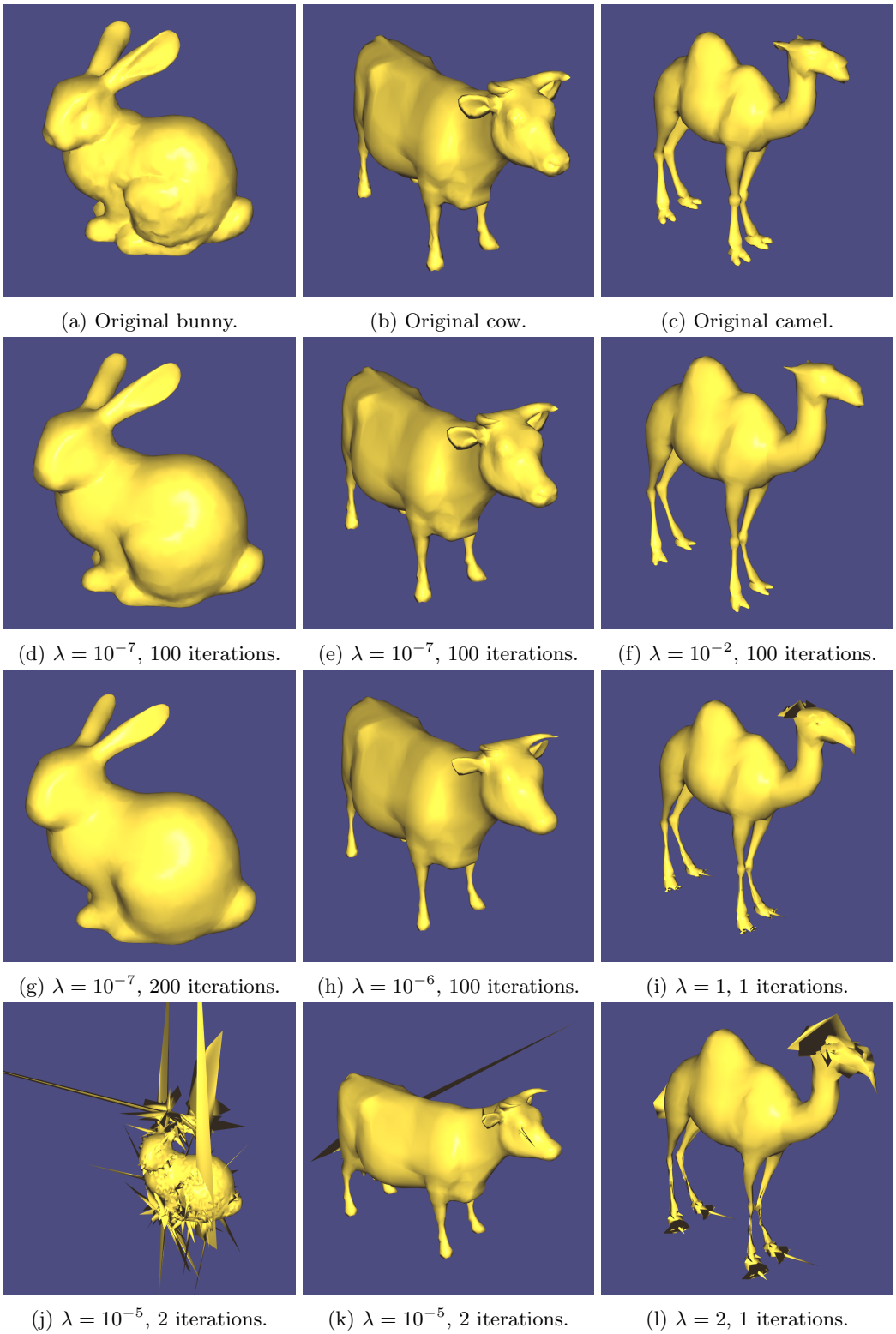
and the scale factor on the menu window is proportional to the dimensions of the bounding box of the mesh, see Fig.12.

Then, in Fig.13 we will show the mesh smoothing results using the implicit form, discussed in task 7. As we can see, the mesh is correctly smoothed, i.e. the spikes are much reduced. The number of iterations and the value of λ affects the final result at different levels. For example, Fig. 13a and 13b differ only by λ and the resulting smoothing is visually sensible. While Fig.13b and 13c differ by number of iterations, and the result is visible as well, but I think not as much as changing λ. Fig.13e and 13f show results from noise level 0.015, using different λ and number of iterations; we can see that doubling λ and halving the number of iterations results in a very similar smoothing.

We can notice from Fig.13d that, once smoothed too much, the details are all lost, almost making the mesh unrecognizable. For this reason, using good values of λ and iterations is important.

3.4 Task 8

Demo Fig.14 shows the bunny mesh smoothing using the implicit form of the LaPlacian, with a fixed value of λ and different number of iterations.



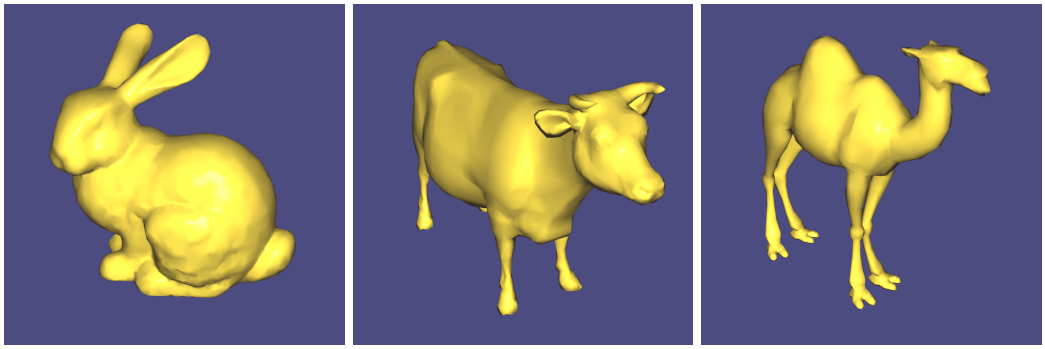
(a) Original bunny. (b) Original cow. (c) Original camel.

(d) $\lambda = 10^{-7}$, 100 iterations. (e) $\lambda = 10^{-7}$, 100 iterations. (f) $\lambda = 10^{-2}$, 100 iterations.

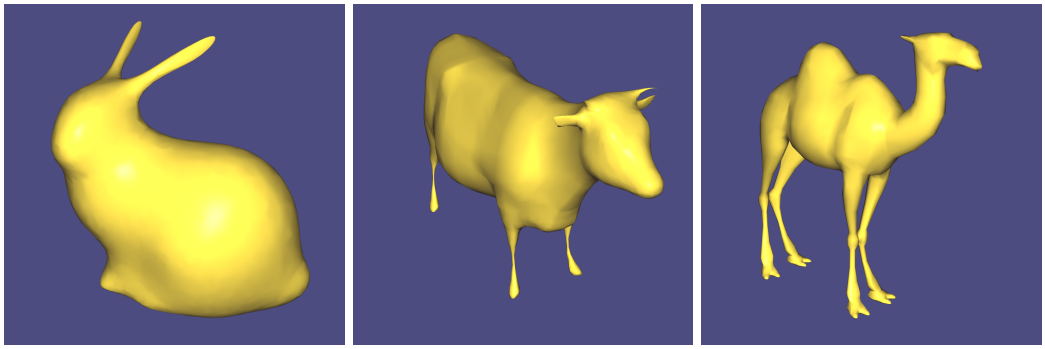
(g) $\lambda = 10^{-7}$, 200 iterations. (h) $\lambda = 10^{-6}$, 100 iterations. (i) $\lambda = 1$, 1 iterations.

(j) $\lambda = 10^{-5}$, 2 iterations. (k) $\lambda = 10^{-5}$, 2 iterations. (l) $\lambda = 2$, 1 iterations.

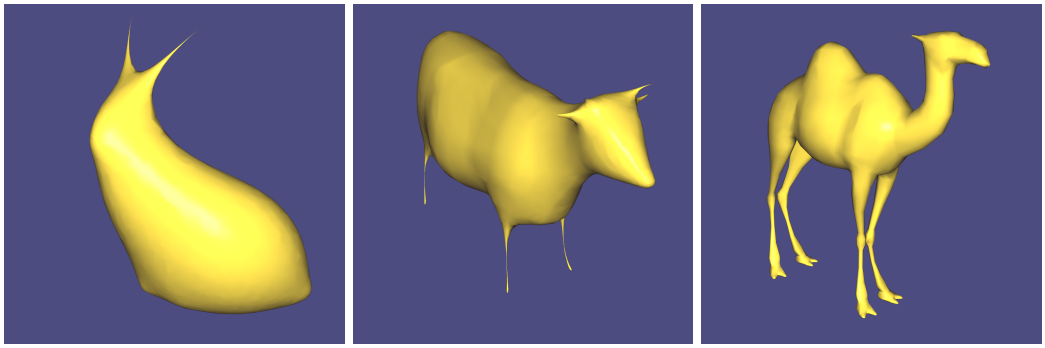
Figure 10: Mesh smoothing using the *explicit* LaPlacian. Different results are shown using different values of λ and iterations.



(a) Original bunny. (b) Original cow. (c) Original camel.

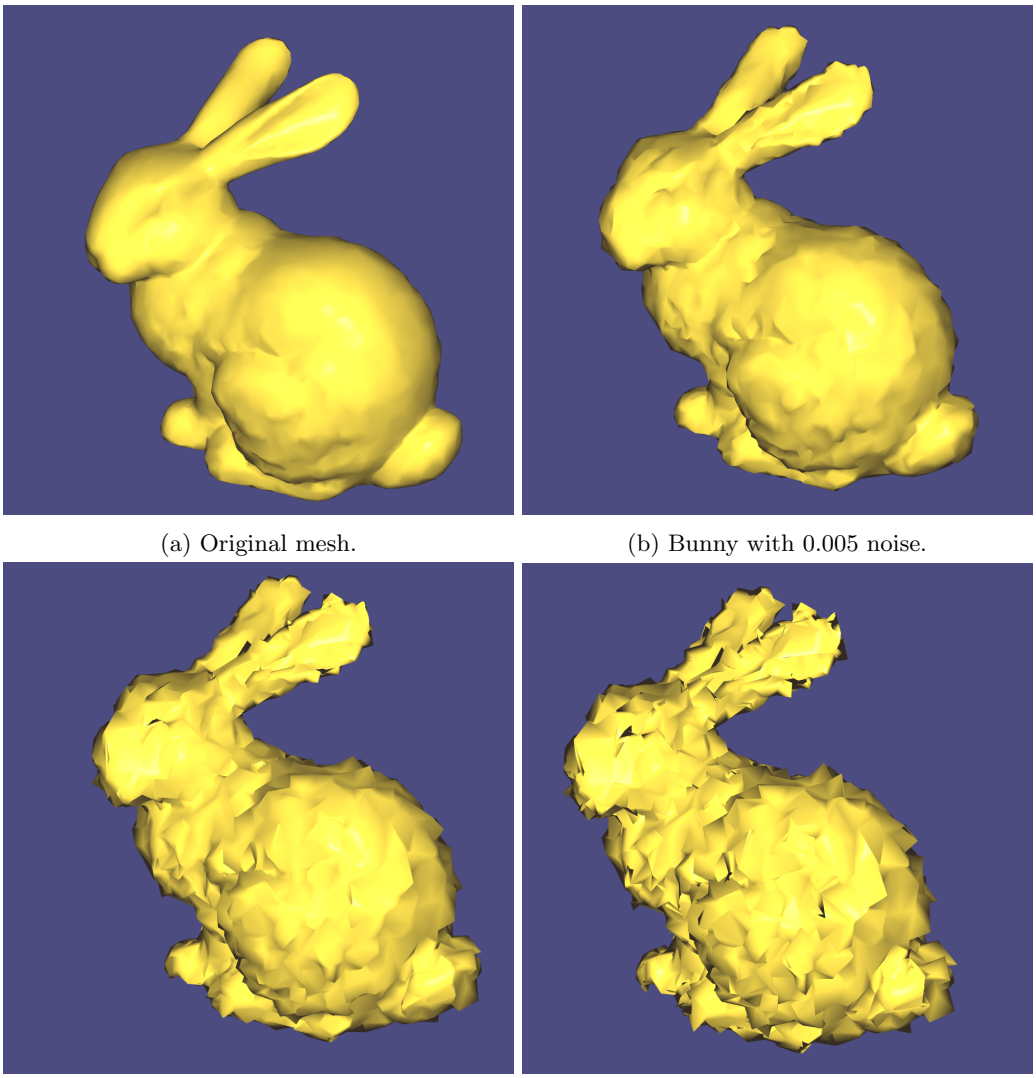


(d) $\lambda = 10^{-6}$, 100 iterations. (e) $\lambda = 10^{-6}$, 200 iterations. (f) $\lambda = 1$, 1 iterations.



(g) $\lambda = 10^{-5}$, 100 iterations. (h) $\lambda = 10^{-5}$, 100 iterations. (i) $\lambda = 2$, 1 iterations.

Figure 11: Mesh smoothing using the *implicit* LaPlacian. Different results are shown using different values of λ and iterations.



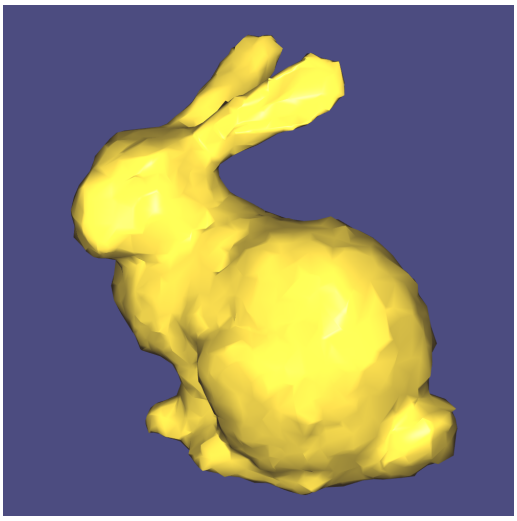
(a) Original mesh.

(b) Bunny with 0.005 noise.

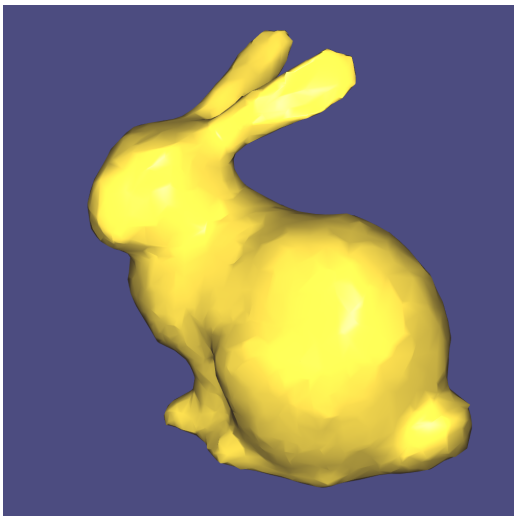
(c) Bunny with 0.010 noise.

(d) Bunny with 0.015 noise.

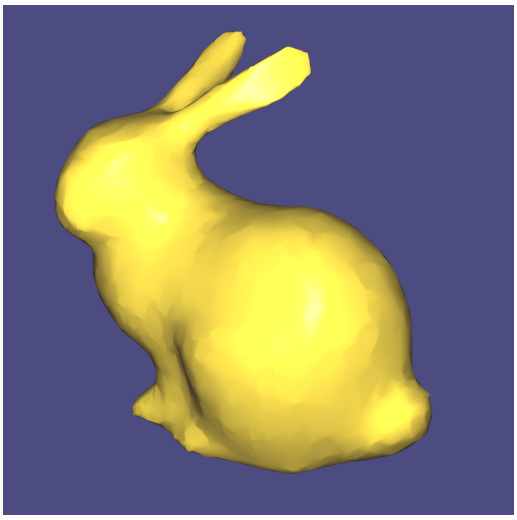
Figure 12: Example of adding Gaussian noise to the bunny mesh. The noise is proportional to the dimensions of the bounding box (bb).



(a) From 0.010 noise, $\lambda = 2E-7$, 50 iterations.



(b) From 0.010 noise, $\lambda = 5E-7$, 50 iterations.



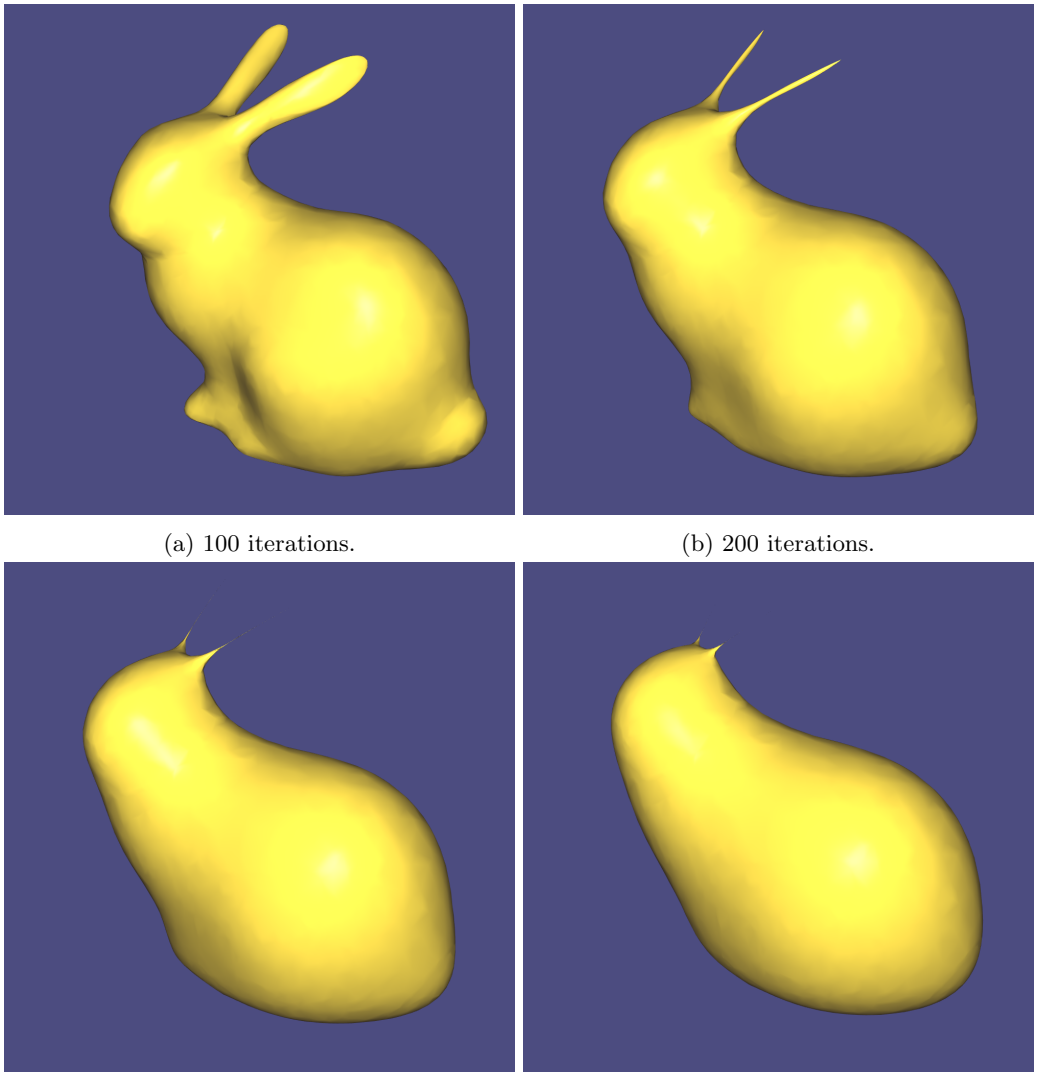
(c) From 0.010 noise, $\lambda = 5E-7$, 100 iterations.

(d) From 0.010 noise, $\lambda = 1E-6$, 200 iterations.

(e) From 0.015 noise, $\lambda = 5E-7$, 100 iterations.

(f) From 0.015 noise, $\lambda = 1E-6$, 50 iterations.

Figure 13: Example of mesh smoothing for the bunny mesh showed in Fig.12.



(a) 100 iterations.

(b) 200 iterations.

(c) 300 iterations.

(d) 400 iterations.

Figure 14: Mesh smoothing using implicit LaPlacian with $\lambda = 5E - 7$.