COMPM080: Coursework #2

Due on Friday, March 20, 2016

Maria Ruxandra Robu- 14042500

Introduction

The section Discrete Curvature introduces different methods to discretize the Laplacian operator on a mesh. The mean curvature H, Gaussian curvature K and the principal curvatures k1 and k2 are implemented and visualized. These have been obtained from both a uniform Laplacian and a Laplace Beltrami operator.

The section Laplacian Smoothing shows results for explicit and implicit smoothing of a mesh on a simple cube and a more complex shape. It also discusses smoothing operations in the context of mesh denoising.

Section 1 - Discrete Curvature

The colour mapping used throughout this section is:

- blue for negative values
- black for values close to 0
- red for positive values

1. Uniform Laplacian and mean curvature

The uniform Laplacian operator is implemented after the equation 1.

$$\Delta f(v_i) = \frac{1}{|N(v_i)|} * \sum_{v_j \in N(v_i))} (v_j - v_i) \tag{1}$$

After the second derivative is computed at each vertex in the mesh, the mean curvature can be approximated. The mean curvature is in the opposite direction of the normal, so the vertex normals are needed for its estimation. The normals have been computed in MeshLab and imported with the mesh.

$$\Delta f = -2Hn \tag{2}$$

$$H(v_i) = \frac{1}{2} < \Delta f(v_i), n(v_i) > \tag{3}$$

A scalar value is obtained for each vertex for the mean curvature. Figure 1 shows the results for the dragon mesh.

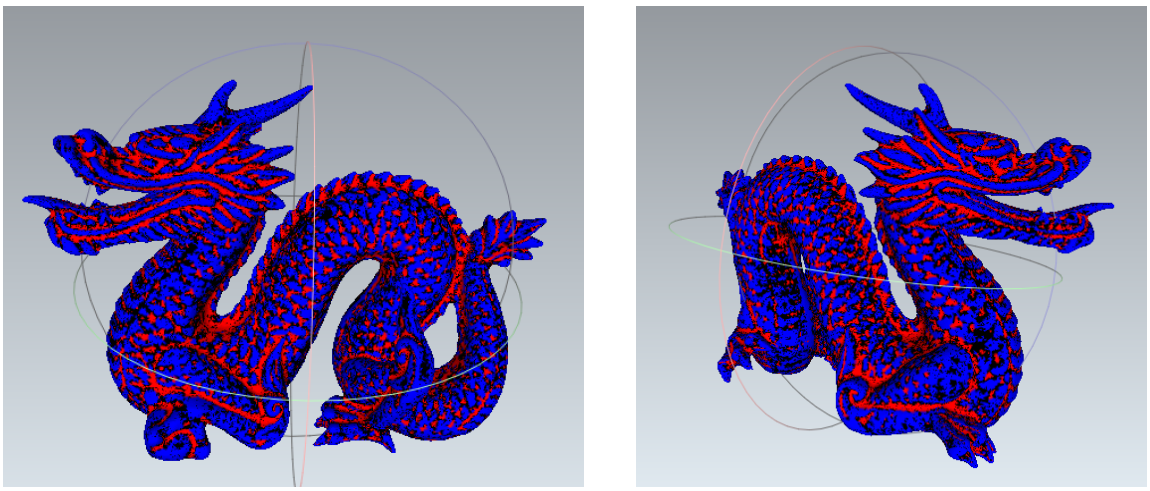
2. Gaussian curvature and the principal curvatures

The Gaussian curvature is obtained by computing the angle deficit around each vertex. Figure 2 illustrated the values for the Gaussian curvature on the dragon mesh. As expected, the red values appear in elliptical areas (tips of scales), whereas the blue ones in hyperbolic areas (in saddles). The areas that can be approximated with a plane locally appear as black, with values close to 0.

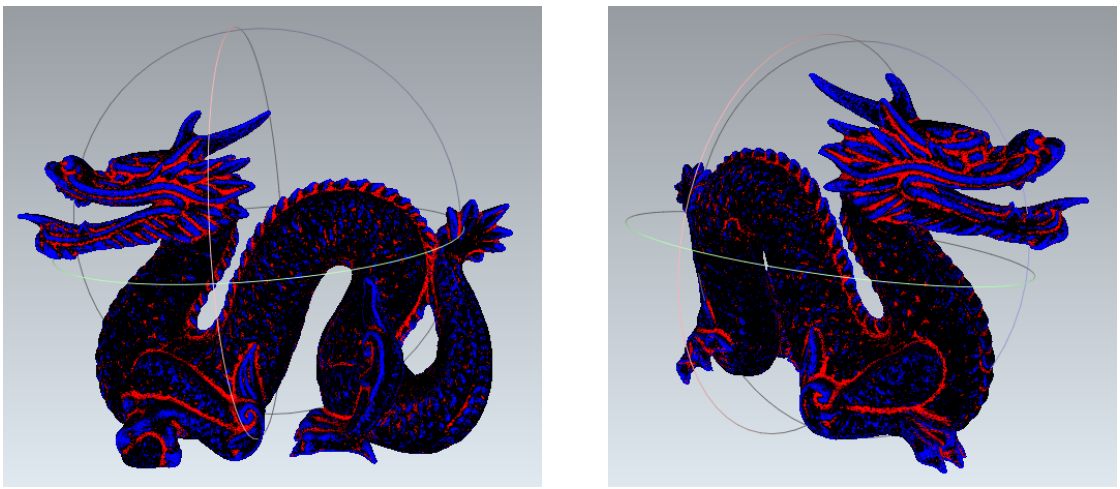
$$K(v_i) = \frac{1}{A(v_i)}(2\pi - \sum_{j \in N(v_i)} \theta_j) \tag{4}$$

Once the mean and Gaussian curvature have been computed, the principal curvatures k1 and k2 can be calculated using equation 7.

$$H = \frac{k_1 + k_2}{2} \tag{5}$$



(a) (b)



(c) (d)

Figure 1: Mean Curvature: Top Row - Uniform Laplace; Bottom Row - Laplace Beltrami

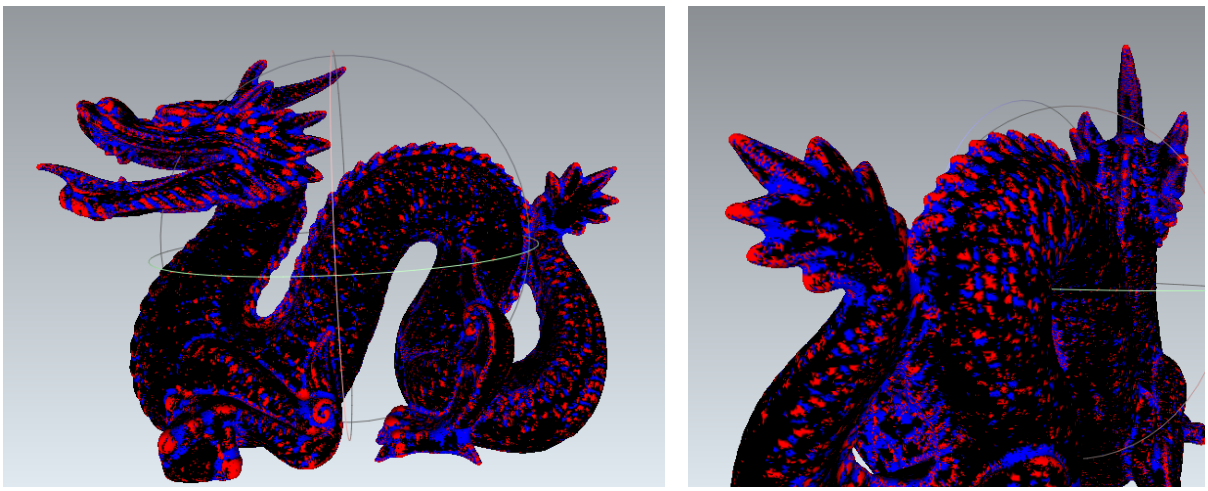
$$K = k_1 k_2 \tag{6}$$

$$k_{1,2} = H \pm \sqrt{H^2 - K} \tag{7}$$

If the difference under the square root is not positive, the principal curvatures are set to 0. Figure 3 shows the results for the principal curvatures obtained through the uniform Laplacian versus the ones obtained with the cotangent discretization. The right column shows a better representation of the surface by introducing more detail.

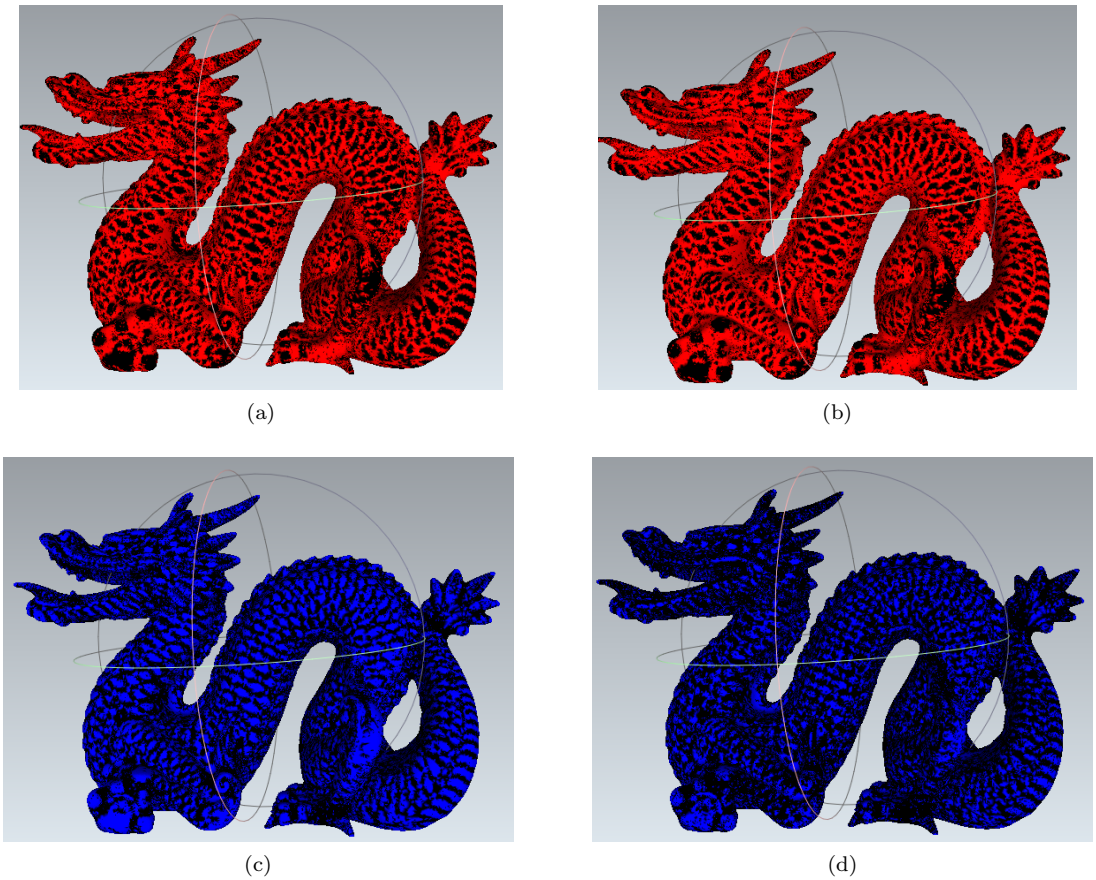
4. Discrete Laplace-Beltrami

The Laplace Beltrami operator is computed using the cotangents of the adjacent angles to a neighbouring vertex as weights.



(a) (b)

Figure 2: Gaussian Curvature



(a) (b)

(c) (d)

Figure 3: Principal Curvatures: a) Maximum curvature - Uniform Laplacian; b) Maximum curvature - Non-uniform Laplacian; c) Minimum curvature - Uniform Laplacian; b) Minimum curvature - Non-uniform Laplacian

$$\Delta_s f(v) = \frac{1}{2 * A(v_i)} * \sum_{v_j \in N(v_i)} (cotan(\beta_i) + cotan(\alpha_i))(v_j - v_i) \tag{8}$$

$$cotan(\alpha) = \frac{< edge1, edge2 >}{||edge1 \wedge edge2||}, where ||edge1|| = ||edge2|| = 1 \tag{9}$$

The operator is normalized by the area of the barycentric cells, which is 1/3 of the area of a neighbouring triangle.

$$A(v_i, v_j, v_{j+1}) = \frac{1}{6}||(v_j - v_i) \wedge (v_{j+1} - v_i)|| \tag{10}$$

The mean curvature was recomputed using the Laplace Beltrami operator and the results are shown in figure 1. The surface is better represented as the angles surrounding a vertex and the corresponding areas are taken into consideration. For example, flat areas have more values closer to 0 (black), as they should. The principal curvatures have been recalculated and the results are shown in figure 3, in the right column.

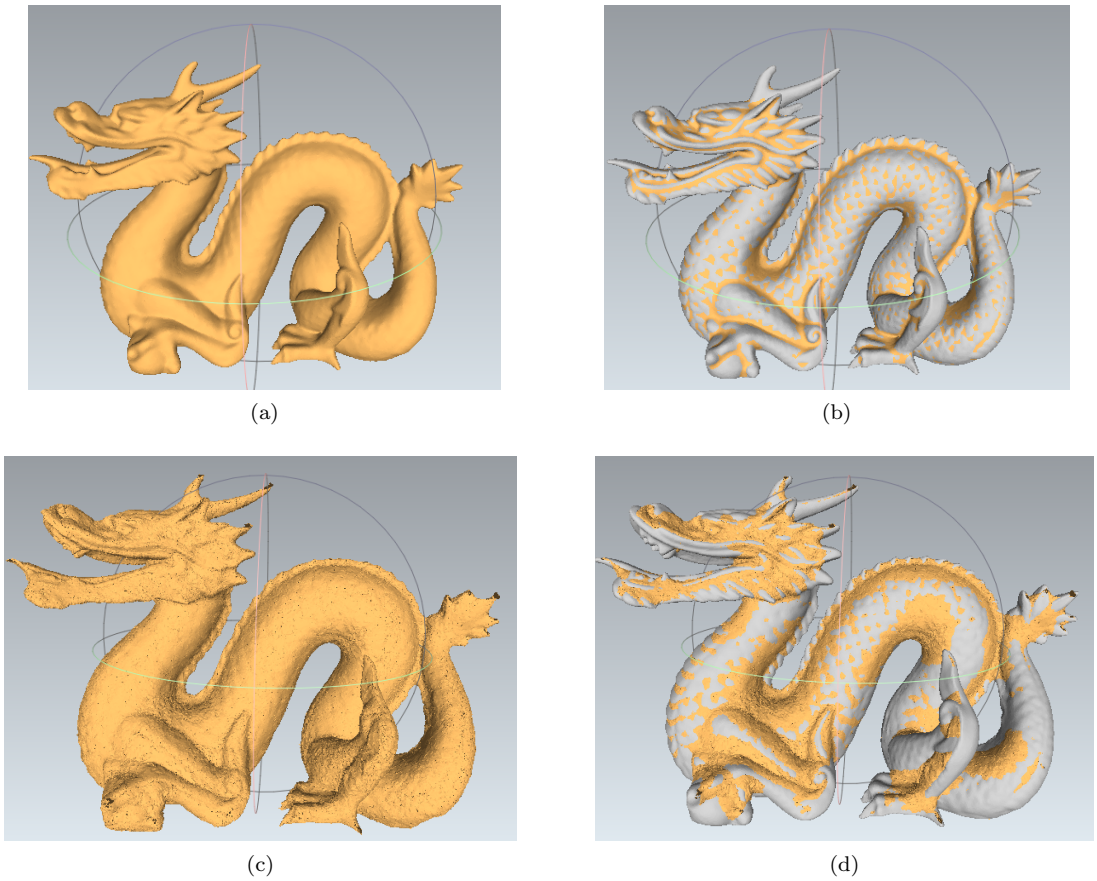
Section 2 - Laplacian Mesh Smoothing

5. Explicit Laplacian Smoothing

Explicit Laplacian Smoothing was implemented using the equation 11, iteratively, for each vertex in the mesh:

$$v_i = v_i + \lambda\Delta(v_i) \tag{11}$$

The results for different lambdas and numbers of iterations are presented in figures 4. From my experiments, values for λ that are bigger than 1 start adding noise to the initial mesh, like in figure 4c.



(a) (b)

(c) (d)

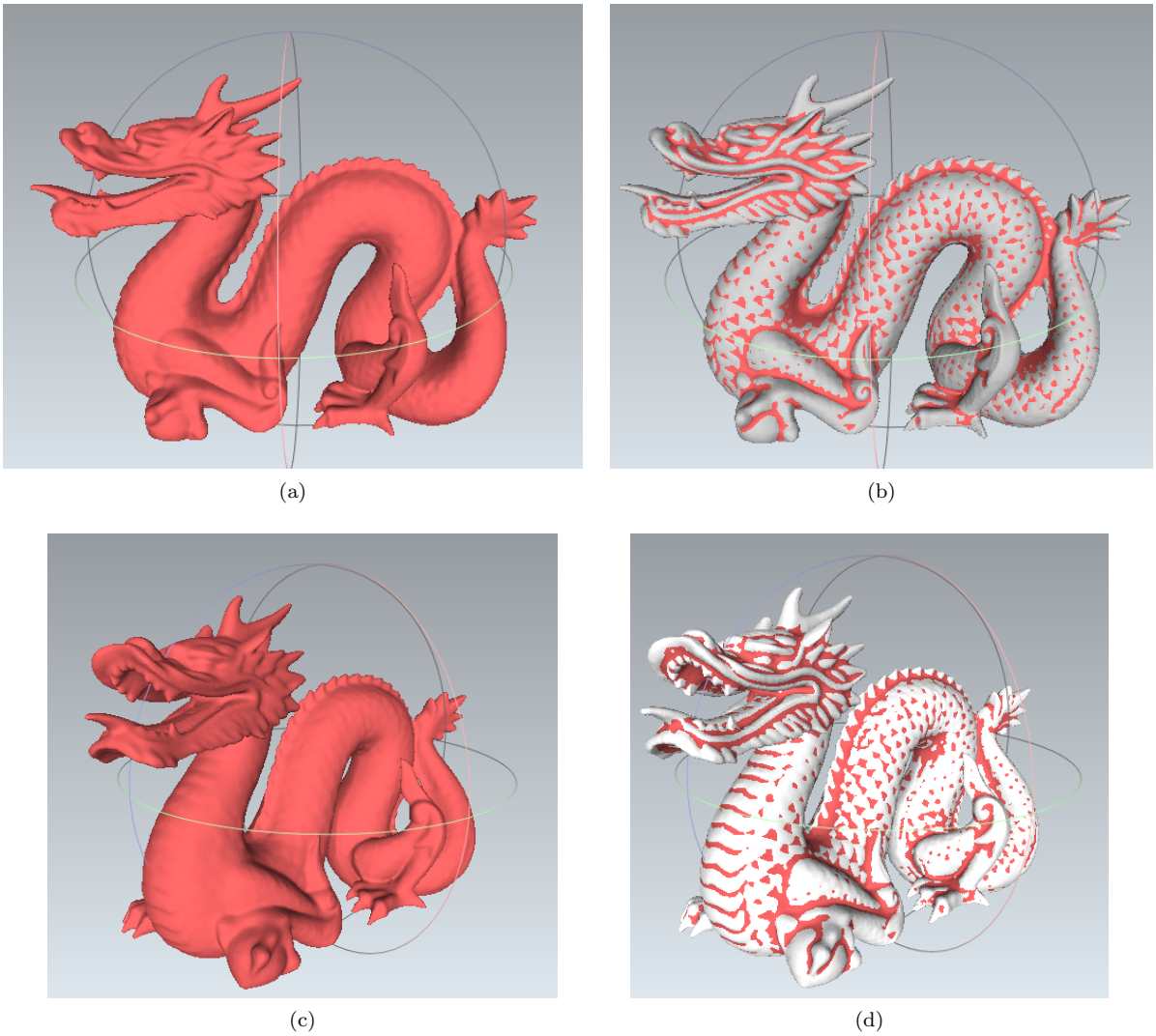
Figure 4: Explicit Smoothing: a) Lambda = 1, Iterations = 5; b) Difference between the smoothed (yellow) and original (white) mesh; c) Lambda = 2, Iterations = 3; d) Difference between the smoothed (yellow) and original (white) mesh

6. Implicit Laplacian Smoothing

This restriction can be removed by implementing implicit smoothing. Instead of increasing the smoothing incrementally in small steps, this formulation allows bigger jumps by choosing higher values for λ.

$$(I - \lambda\Delta)v_{i+1} = v_i \tag{12}$$

The implicit smoothing equation was solved using the bi conjugate gradient stabilized solver (Eigen::BiCGSTAB). Figure 5 shows results for various setting. As opposed to the explicit smoothing, this implementation does not introduce noise and allows for a more stable result.

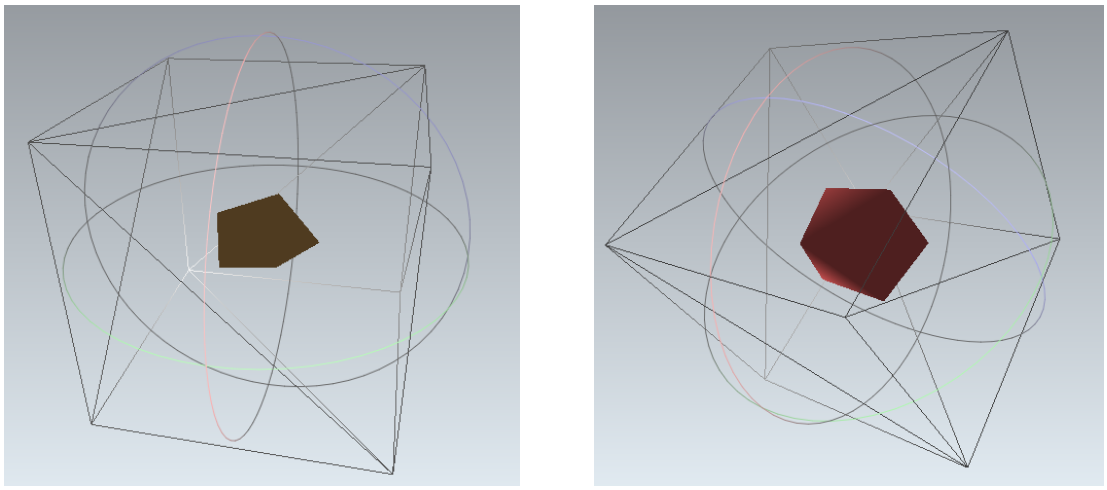


(a) (b) (c) (d)

Figure 5: Implicit Smoothing: a) Lambda = 5, Iterations = 1; b) Difference between the smoothed (pink) and original (white) mesh; c) Lambda = 3, Iterations = 3; d) Difference between the smoothed (pink) and original (white) mesh

These smoothing algorithms have been applied to both simple and complex meshes. Figure 6 shows the results for a cube. Only the edges have been shown from the original mesh, to provide a comparison to the smoothed cubes. In this case, λ has to have even smaller values for the explicit smoothing. If $\lambda = 1$, the cube gets very small and it loses its shape.

These smoothing algorithms have been implemented based on the uniform discretization of the Laplacian. This implementation choice is obvious in the context of the cube, where smoothing makes the object deform. A better approach would be to use the cotangent discretization. Since the Laplace Beltrami operator incorporates information about the normals, the smoothing would preserve more of the initial shape of the objects.



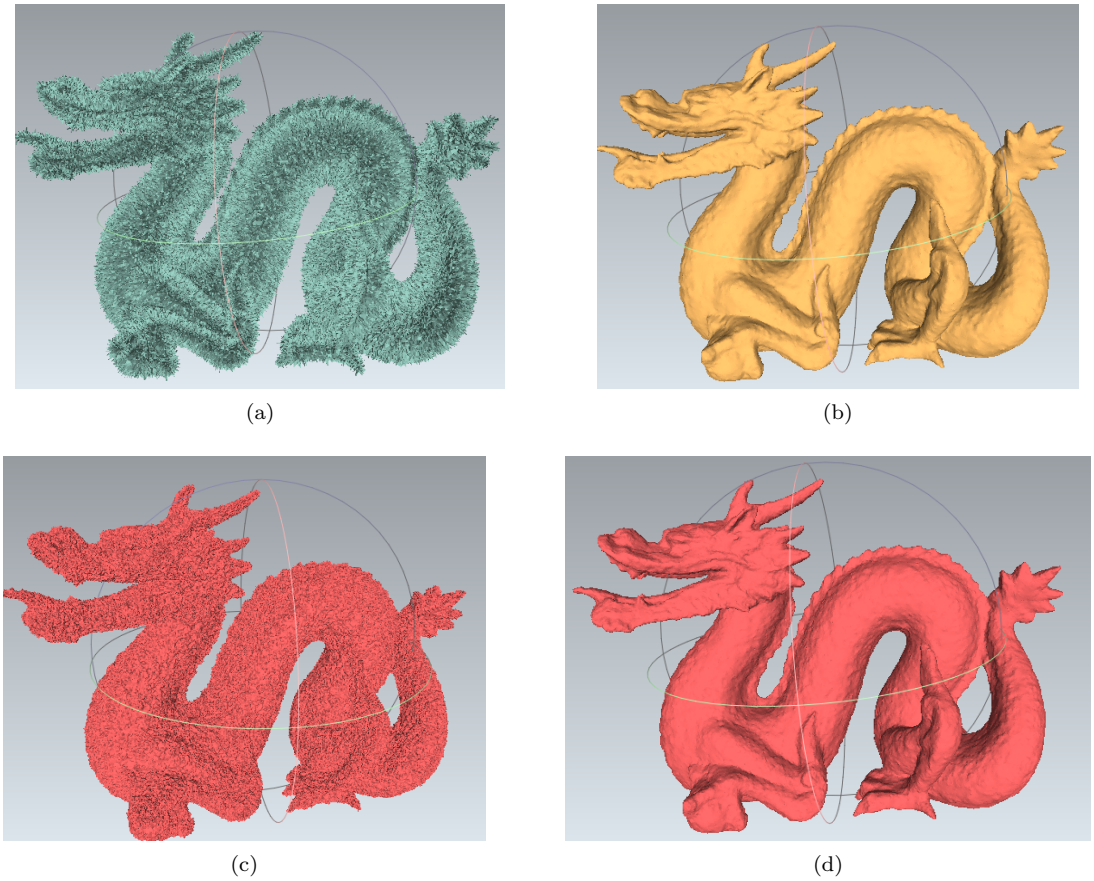
(a) (b)

Figure 6: Cube - Explicit vs. Implicit Smoothing: a) Lambda = 0.5, Iterations = 3; b) Lambda = 3, Iterations = 3

Section 2 - Laplacian Mesh Smoothing continued on next page...

7. Denoising experiments

Gaussian noise with mean 0 and standard deviation of 0.001 was added to the dragon mesh. The challenge in using Laplacian smoothing to denoise a mesh is in removing the noise whilst at the same time, keeping the high frequency details. Figures 7 and 8 show some results for different parameters. Lambda was chosen to be lower than 1 in all the experiments with explicit smoothing, since higher values add more noise. Figures 7 b) and d) show a smooth mesh, with loss in the details around the scales and the head of the dragon.



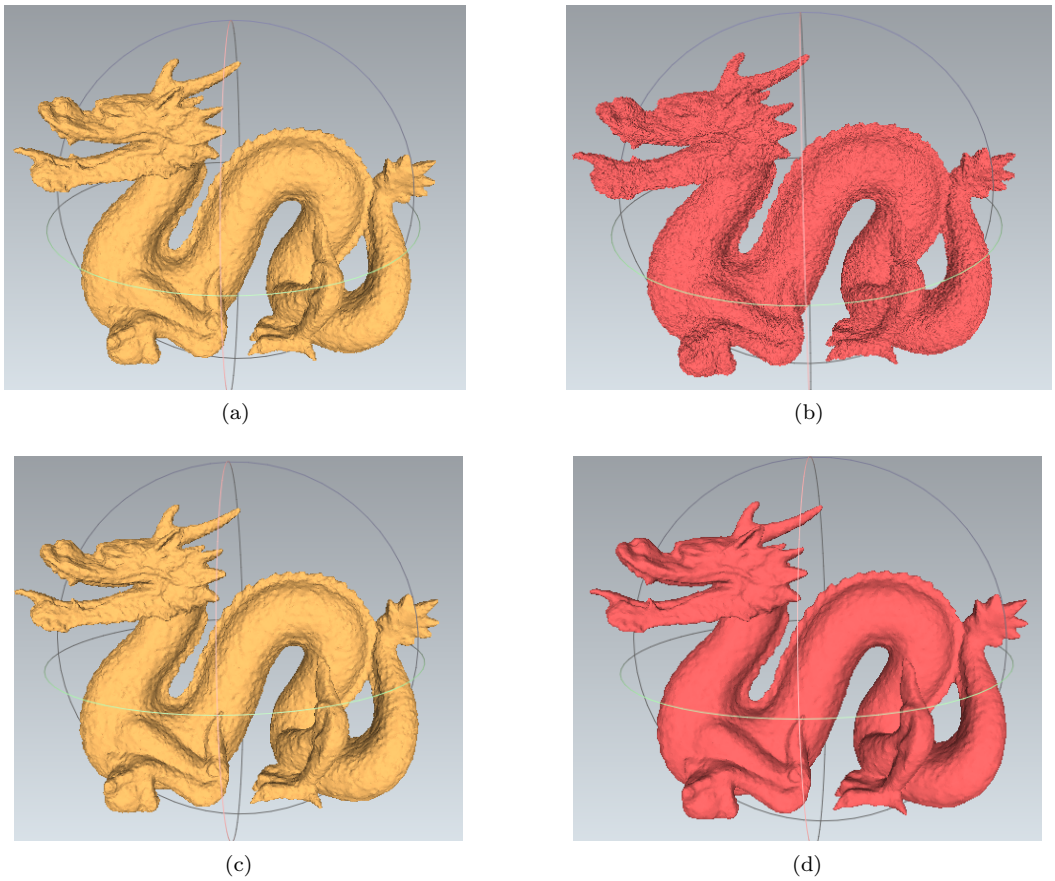
(a) (b)

(c) (d)

Figure 7: Experiments with noise: a) Noise with std = 0.001 was added to the dragon mesh; b) Explicit Smoothing - $\lambda = 1$, iterations = 5; c) Implicit Smoothing - $\lambda = 3$, iterations = 1; d) Implicit Smoothing - $\lambda = 3$, iterations = 3

In order to decrease the loss of details, further setting were tried for the smoothing. Figures 8a) and b) show that examples in which the noise is still present after smoothing.

In conclusion, depending on the amount of noise present, Laplacian smoothing can be used to retrieve the original shape, to some extent. Experiments have to be conducted in order to find the optimal parameters that would recover enough of the high frequency details, whilst removing a reasonable amount of noise. From the above experiments, lower values of lambda with a higher number of iterations lead to better results. However, this would increase the computation time since each iteration goes through the whole mesh.



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

Figure 8: Experiments with noise: a) Explicit Smoothing - $\lambda = 1$, iterations = 3; b) Implicit Smoothing - $\lambda = 2$, iterations = 2; c) Explicit Smoothing - $\lambda = 0.7$, iterations = 5; d) Implicit Smoothing - $\lambda = 2$, iterations = 5