TP 5: Surface Reconstruction

Objectives

- Test 3D Surface Reconstruction on Meshlab
- Surface Reconstruction on Python : implement the Hoppe/IMLS implicit function

A. 3D Reconstruction on Meshlab

The goal is to test and understand the two main surface reconstruction methods (RIMLS and Poisson) on two point clouds: “bunny.ply” and “dragon.ply”.

1) You need to install Meshlab v2016 (or newer)

2) First, you compute the normals with “Filters->Point Set->Compute normals for point sets” (the parameter “Neighbour sum” is the k of the k nearest neighbors for the PCA computation, 10 is good for dense point clouds with little noise)
   Hint: you can render the normals with “Render->Show Normal”. To get a mesh with an orientation toward exterior, you need normals with orientation toward exterior. You can invert the orientation of the normals with “Filters->Normals, Curvature and Orientation->Per Vertex Normal Function”

3) Then, you can test the two reconstruction methods seen in the course : RIMLS and Poisson :
   a) RIMLS: “Filters->Remeshing, Simplification and Reconstruction->Marching Cubes (RIMLS)”. The two main parameters are the MLS filter scale and the grid resolution
   b) Poisson: “Filters->Remeshing, Simplification and Reconstruction->Screened Poisson Surface Reconstruction”. The main parameter is the reconstruction depth.

<u>Question 1:</u> Take screenshots of the two reconstructed meshes for the two point clouds (with best possible parameters). In total, you should have 4 screenshots.
Hint: To find best parameters, you need to define what best is for a surface reconstruction method: usually it is a good trade-off between geometric details of the surface, minimum unwanted holes and minimum vertices of the mesh

<u>Question 2:</u> For each point cloud, what is the “best” surface reconstruction method? (give the parameters used for that reconstruction, the final number of vertices and faces)

B. Surface Reconstruction on Python: implement the Hoppe implicit function

The goal is to compute the hoppe implicit function on a regular grid. Then, you will be able to extract the surface as a mesh from the iso-zero.

You need as input, point clouds with normals. You can compute the normals using CloudCompare or your code from previous TP (be careful, you need the good orientation of the normals).

1) To compute the hoppe function, we create a regular grid of the space around the input point cloud.
2) Then, on every node x of the grid, the hoppe function is $f(x) = n_i \cdot (x - p_i)$ when p_i is the closest point of the point cloud to x (and n_i the associated normal of point p_i).

The result is a scalar field on a regular grid. To visualize your result and extract the iso-zero, you have the Python package called "scikit-learn" with Marching Cubes algorithm

You can then view the mesh result with Matplotlib.

Question 3: Take a screenshot of your iso-zero surface of the sphere point cloud

Question 4: Take a screenshot of your iso-zero surface of the bunny point cloud

C. Implement the EIMLS function (BONUS)

We have seen in the course that Hoppe implicit function is not a continuous surface representation.

The Implicit Moving Least Square (IMLS) function is a good approximation of the distance of any point to the surface when the point is close enough to the surface.

For any point x, the IMLS function $f(x)$ is defined by :

$$f(x) = \frac{\sum_i \ (n_i \cdot (x - p_i))\theta(x - p_i)}{\sum_i \ \theta(x - p_i)}$$

with $\theta(x - p_i) = e^{-\frac{||x-p_i||^2}{h^2}}$

The parameter h is proportional to the noise of the point cloud. $h = 0.003$ is a good trade-off for the bunny point cloud.

You can compute the function $f(x)$ using all points p_i of the point cloud but you can see that the weights $\theta(x - p_i)$ decrease quickly when p_i are far from the point x. It is more efficient to compute the function using only the k nearest points around x ($k = 10$ is enough for the bunny point cloud).

You will have an issue when the point x is far from the surface: the exponential weights will become null because of numerical limits of floats representations on computers.

To deal with that, we define the Extended Implicit Moving Least Square (EIMLS), the parameter h will vary following x (with p_i the closest point of the point cloud to x):

$$h = max(0.003, \frac{||x - p_i||}{4})$$

For more details about the EIMLS, you can search for Hassan Bouchiba Thesis.

<u>Question 5: Show with screenshots the differences between the Hoppe surface and the IMLS/EIMLS surface of the Bunny? Where does it comes from?</u>