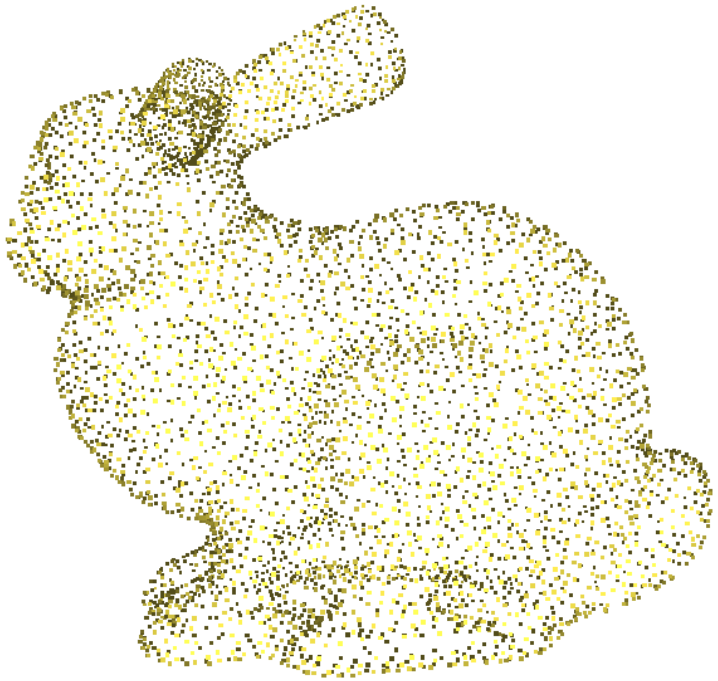
Project 2: Surface reconstruction

Goal

- Implement a program in Java for reconstructing a surface from a 3D point-cloud, and visualizing the surface



Input 3D point-cloud

Reconstructed surface

Tasks

- Load a 3D point-cloud using the xyz file format (list of point coordinates and normal)
- Visualize the 3D point-cloud using OpenGL
- Implement surface reconstruction with radial basis functions (RBF)
- Implement the Marching Cubes algorithm to convert the zero level-set of the RBF to a triangle mesh
- Implement data-structures for manipulating triangle meshes
- Visualize the reconstructed surface using OpenGL

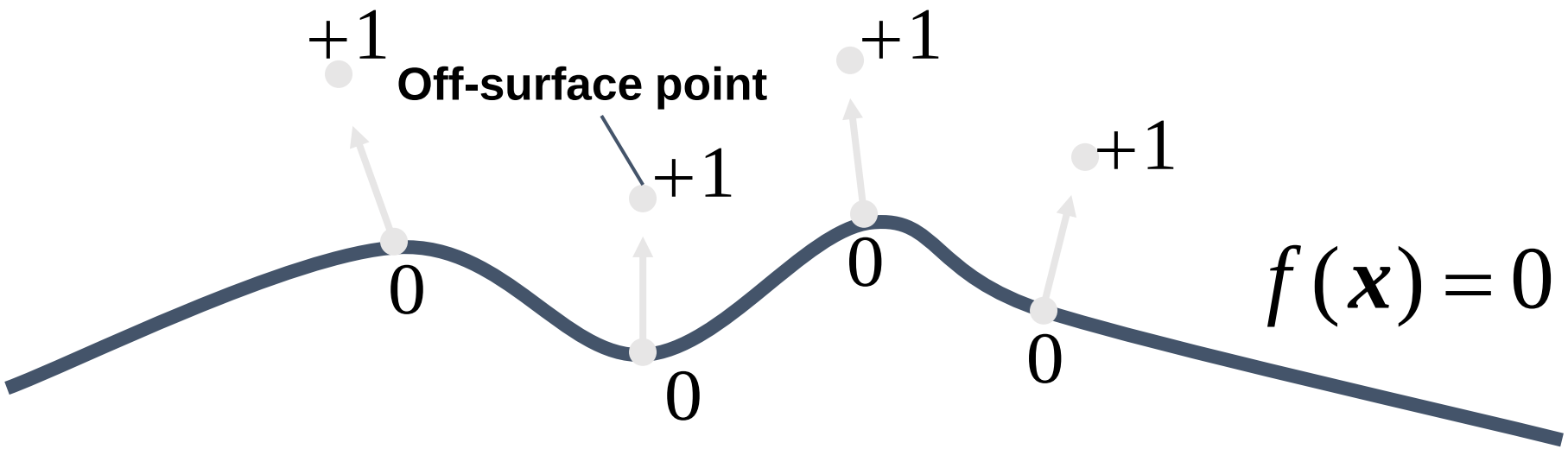
Input: 3D point-cloud

Input 3D point-cloud

coordinates | normals

-0.256436 0.583325 0.067527 0.481354 -0.272355 0.833139
0.318937 -0.915439 0.424017 -0.486736 -0.870114 -0.077389
-0.301311 -0.982578 0.040985 0.084952 -0.993386 -0.077247
0.326375 -0.969928 0.508926 -0.117271 -0.968891 -0.217940
0.282044 -0.963641 0.507701 -0.141128 -0.904927 -0.401486
-0.485513 -0.661363 0.305646 -0.423470 -0.847007 0.321328
-0.146155 -0.961498 -0.234851 0.025947 -0.682755 0.730187
-0.137954 -0.984448 -0.287432 0.011215 -0.996068 0.087883
-0.961686 0.409485 0.237700 -0.978617 -0.083478 -0.187988
0.025256 -0.968669 0.597865 -0.070133 -0.953594 -0.292815
0.045558 -0.959368 0.573894 -0.125330 -0.848953 -0.513392
-0.017551 -0.965747 0.600896 -0.017954 -0.951261 -0.307863
-0.481140 -0.972196 0.456415 0.009993 -0.999653 0.024375
-0.222935 -0.968050 0.480761 0.355087 -0.920759 -0.161603
…

RBF reconstruction



$$\hat{f}(\boldsymbol{x}) = p(x) + \sum_{i=1}^{n} \lambda_i \varphi\left(\left\|\boldsymbol{x} - \boldsymbol{x}_i\right\|\right)$$

$p(x)$: Linear polynomial

$\varphi(r) = r$

Mesh data-structure

- Indexed face list
- Keep two lists:
  - List of vertex coordinates
    V = [x0, y0, z0; x1, y1, z1; …]
  - List of vertex indices per face
    F = [0, 1, 2; …]

For example: ArrayList < Vec3f > for the vertices, and ArrayList < Tuple3i > for the triangles.

Additional lists for the normals (per vertex, per face), and eventually dictionaries for the connectivity information (if needed).