1 Steps

1. **Computing the non-oriented normals**.
   - **Computing the center of gravity**. The center of gravity is computed by the function, **center–of– gravity**.
   - **Computing the matrix A**. Matrix A is defined
   $$A = pts - B,$$
   where, pts is a list of K neighbors and B is the coordinate of the center of gravity. Next, three vectors a_1, a_2, a_3 with k components defined as the columns of matrix A. Then, the matrix $A^T A$ is obtained by scalar product between these three vectors. Finally, the eigen vector associated with minimum eigen vector is used as the normal.

2. **Minimum spanning tree**. For this case, at first, the function **computeRadius** is defined to find the appropriate radius. Then for every point p_i, the other points p_j, are checked to determine which points belong to the circle with radius r. Next the weight is computed by
   $$1 - || < n_i, n_j > ||,$$
   Where n_i and n_j are the normal vectors at point p_i and p_j respectively, and $| < \cdot, \cdot > |$ denotes the absolute value of the scalar product. The weighted arc is added by the related function. Finally the function **arbre_couvrant_minimal()** is called on the proximity weighted tree so it will return the minimal spanning tree.

3. **Reorientation of normals**. In the function computeOrientedNormals(), first we set a vector of boolean that tells if a node has been browsed or not. Then the recursive function **normalsRedirection()** is called. In this function, for every node sons , we look if the scalar product between its normal and its father's normal is negative. If it is, that means the normals of the father and the son are in opposite direction so we change the direction of son's normal. Then we call the recursive function on the son.

4. **The implicit function**.
   For every point of the mesh, we compute the implict function
   $$f(x) = \frac{\sum n_i^T (x - x_i) wi(x)}{\sum w_i(x)}$$

5. **Isovalue surface**. First we computed the 3D Grid delimited by the minimum and maximum x,y and z of the points of the mesh. Then we created a vector v of size $(nx + 1)$ x $(ny+1)$ x $(nz+1)$, with nx x ny x nz as the resolution of the 3D grid. And for every index of that array we assigned the value of the implicit function that we coded previously. Finally, we computed the isovalue surface by calling the given function **surface_isovaleur()**

6. **Final normals**.
   The final normals n are defined for each vertices of each triangles in surfacep by the following equations :
   $$\begin{aligned} nx &= f(x_i - 0.01, y_i, z_i) - f(x_i + 0.01, y_i, z_i) \\ ny &= f(x_i, y_i - 0.01, z_i) - f(x_i, y_i + 0.01, z_i) \\ nz &= f(x_i, y_i, z_i - 0.01) - f(x_i, y_i, z_i + 0.01) \\ n &= normalize(n_x, n_y, n_z) \end{aligned}$$