CS468 - Geometry Processing Algorithms

Exercise 5 - Remeshing

Handout date: 11/29/2010

Submission deadline: 12/12/2010, Midnight

Note

Copying of code (either from other students or from external sources) is strictly prohibited! Any violation of this rule will lead to expulsion from the course.

What to submit

A .zip compressed file renamed to "Exercise*n*-YourName.zip" where *n* is the number of the current exercise sheet. It should contain:

- A Microsoft Visual Studio 2008 file solution with all source files and project files. In order to avoid sending build files, close your Visual Studio solution and run the file "cleanSolution.bat" located in the top directory of the framework before you submit.
- A "readme.txt" file containing a description on how you solved each exercise (use the same numbers and titles) and any problems encountered.
- Send your solutions to mirela@stanford.edu before the submission deadline.
  No late submissions will be accepted.

Reference Software

See 05-Remeshing.exe for an example of how your program should run.

Remeshing

In this exercise you will implement a surface-based remeshing algorithm composed of four main steps:

- Split long edges
- Collapse short edges

- Flip edges to improve vertex valences
- Tangential smoothing to improve triangle quality

Framework

A new project, Remeshing has been added to the framework from the previous exercise. It contains one single file and it compiles to a command line tool that reads a mesh and outputs a remeshed version of it. The program takes three parameters: the target edge length (as a fraction of the model's bounding box diameter), the input mesh file (.off) and the output mesh file (.off).

5.1 Splitting long edges

Implement the split_long_edges(float _long) function so that it splits edges longer than _long in two halves. The loop tries to split edges until no longer edge is present in the mesh, or a maximum of 100 iterations have been executed. Similarly all subsequent tasks will use this limit to make sure the algorithm terminates and does not run for an unreasonably long time.

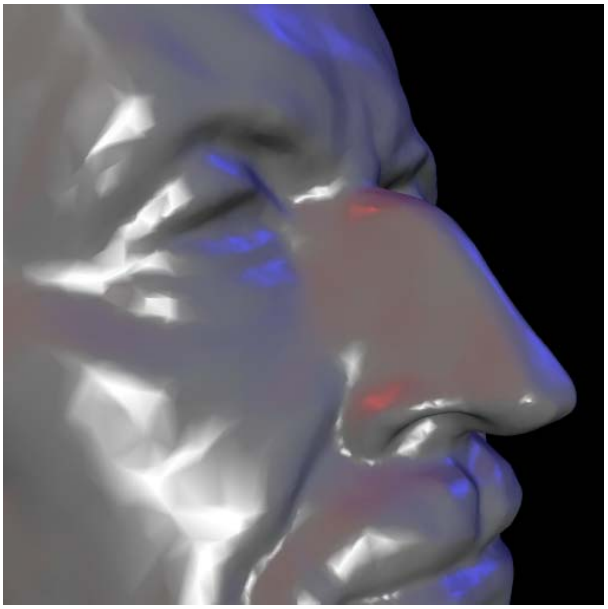
5.2 Collapsing short edges

Complete the collapse_short_edges(float _short) function. First, you should not consider edges connecting a boundary and a non-boundary vertex for collapse to avoid shrinking around the boundaries. Then, you should check if the edge is shorter than the parameter _short. If so, check if both of the halfedges corresponding to the edge are collapsible. If they are, you should collapse the lower valence vertex into the higher one. Otherwise collapse the halfedge which is collapsible, or don't collapse at all if both of the tests fail.

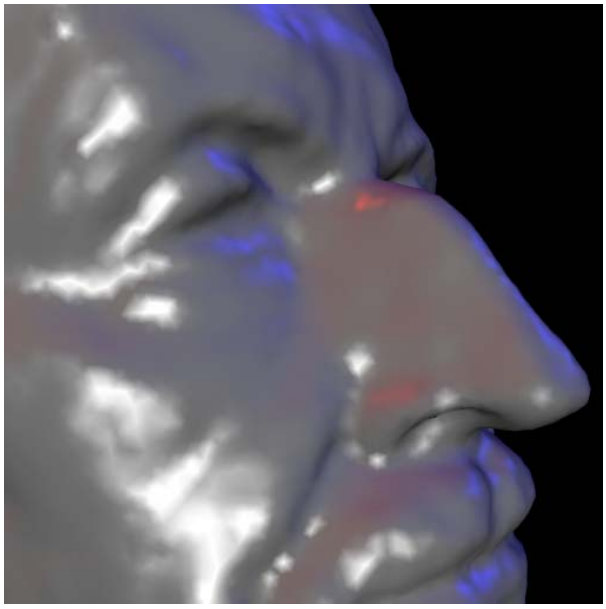
Hint: The is_collapse_ok() function checks if a halfedge can be collapsed.

5.3 Flipping edges to improve valences

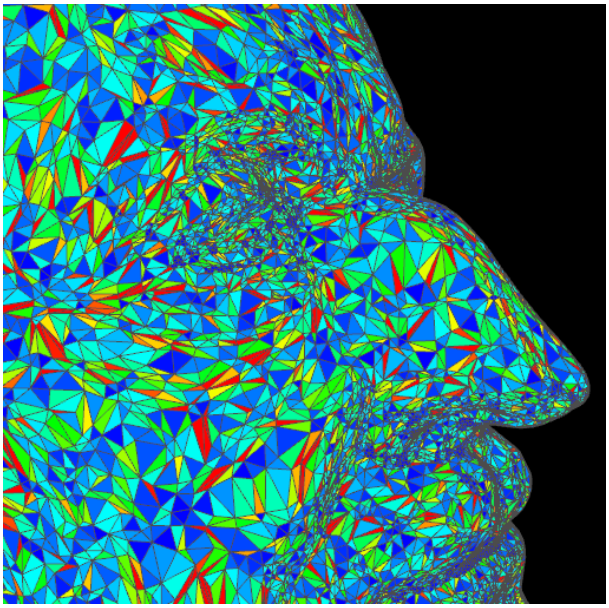
Complete the equalize_valences() function, so that it flips vertices if it improves vertex valences in the local neighborhood.



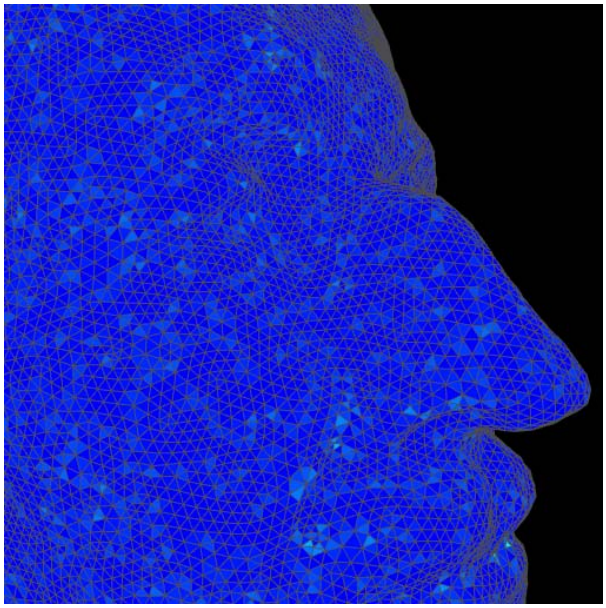
(a) Original with smooth shading



(b) Remeshed with smooth shading



(c) Triangle qualities of original



(d) Triangle qualities of remeshed

Figure 1: Meshes before and after remeshing (using the 06-Remeshing.exe 2.00 max.off max regular.off command)

We know that an "ideal" mesh vertex has valence 4 if it lies on the boundary and 6 otherwise. For an edge *e* now consider the two end-vertices and the two other vertices on the triangles incident on *e*. For every vertex compute its valence deviation, i.e. the difference between the current valence and the optimal valence for this vertex. By comparing the sum of squared valence deviations before and after a potential edge flip you can decide if the edge flip will

improve the valences locally (smaller valence deviations are better). If the edge improves on the local valences, flip it.

Don't forget to use the is_flip_ok() function in order to make sure an edge can be flipped before you try to flip it.

5.4 Tangential smoothing

Implement the tangential_relaxation() function to improve the triangle shapes by smoothing vertices in the tangent plane of the mesh.

Similarly to a previous exercise (Smoothing), approximate the mean curvature with the uniform Laplacian. Now, decompose this vector into two components: one parallel to the vertex normal and one parallel to the tangent plane in the vertex (perpendicular to the normal). Use the tangential component to move the vertex and thus improve the triangle quality. For more details of tangential smoothing consult the lecture notes and the smoothing exercise spec.

As a final result the four remeshing steps should lead to results shown on the Figure 1. This is visualized using the same mechanism as in the Smoothing exercise.

5.5 Bonus (10%)

Implement an adaptive version of the remeshing algorithm based on curvature. The general idea is to encourage splits on edges having high curvatures, and discourage splits of edges with lower curvature.

We give you as hints two ways to approximate/use the curvature for this task:

Use the values of the maximum curvatures at the endpoints of an edge to decide if the edge should be split. To approximate the maximum curvature you can use curvature approximation methods already implemented in previous exercises. Given the mean curvature H and the Gaussian curvature K, the principal curvatures are:

$$k_{max} = H + \sqrt{H^2 - K}$$
$$k_{min} = H - \sqrt{H^2 - K}$$

Another option is to use the curvature on an edge to decide if splitting should happen. You can derive a simple formula of a curvature approximation along an edge. It uses as input the endpoint coordinates of the edge and the corresponding normals. The curvature is the inverse of the radius of the osculating circle. You can approximate the osculating circle by fitting a circle passing through the two endpoints of the edge so that the normals at the vertices are perpendicular to the circle tangent lines at those points. The inverse of the radius of this circle approximates the curvature along the edge.