Digital Geometry Processing
Exercise 5 - Mesh Decimation

Handout date: 03.13.2014
Submission deadline: 03.26.2013, 23:59 h

Note

Copying of code (either from other students or from external sources) is strictly prohibited! Any violation of this rule will be reported to USC Office of Student Judicial Affairs and Community Standards.

What to hand in

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

- All the source code necessary to compile your application. All the source code necessary to compile your application. Please, do not hand in libraries, build folders, binaries. You only need to submit `cc` and `hh` files.
- A "readme.txt" file containing a description on how you solved each exercise (use the same numbers and titles) and the encountered problems.
- Upload the archive at `http://www.dropitto.me/usc-cs599dgp` before the deadline. Password is `ididit`.

Mesh Decimation

In this exercise you will implement surface simplification using quadric error metrics. The following tasks you will implement to complete the framework to a functional mesh simplification algorithm.

- Computing initial quadrics for vertices
- Testing edge contraction for normal flips of triangles
- Computing the collapse priority as evaluation of the sum of two quadrics
- Implementing the global decimation loop: get best halfedge, collapse, update neighborhood

The Framework

A new project, `Decimation` has been added to the framework from the previous exercise. The executable takes three input arguments: decimation rate, the input mesh file (`.off`) and the output mesh (`.off`).

To load the cow mesh

Set **Command Arguments** to: `0.1 data/cow.off data/output-mesh.off`

`output-mesh.off` is the final decimated mesh file name and the first number is the percentage of decimation (in this case 10%).

5.0 Preparation

Read the Garland, Heckbert: Surface Simplification Using Quadric Error Metrics paper to understand all the details of the simplification algorithm.

Our implementation will be a slightly different, simplified version. Differences are the following:

- We are considering only vertex pairs to be contracted *only* if they are connected by an edge
- For every vertex we choose the minimum priority of the adjacent halfedges as the priority of the vertex. We save as the collapse *target* of this vertex the other endpoint of the minimum halfedge. Then put vertices in the queue instead of pairs. Note that at this stage every vertex has associated a priority and a target vertex where it will be contracted if selected for contraction.
- In the main loop of the algorithm we choose always the vertex with minimum priority for contraction. Then, we use the properties attached to the vertex to test and apply the contraction if needed. Contracting a halfedge in our implementation means contracting the source vertex to the target vertex.

5.1 Quadric from triangle

Complete the `init()` function in `deci.cc` so that it calculates vertex quadrics from the quadrics of the incident triangles. Remember, the quadric associated to a vertex is the sum of the quadrics of the triangles. Use the `Quadricd` class.

5.2 Collapse testing with normals

The `is_collapse_legal()` function tests if an edge can be contracted. OpenMesh can test if topologically the edge collapse is possible. You should test if the edge collapse would lead to triangle flipping. To do this, go through every triangle adjacent to the vertex which moves during the collapse. For every triangle compare the original normal with the normal which would result after the collapse. If any of the normal changes

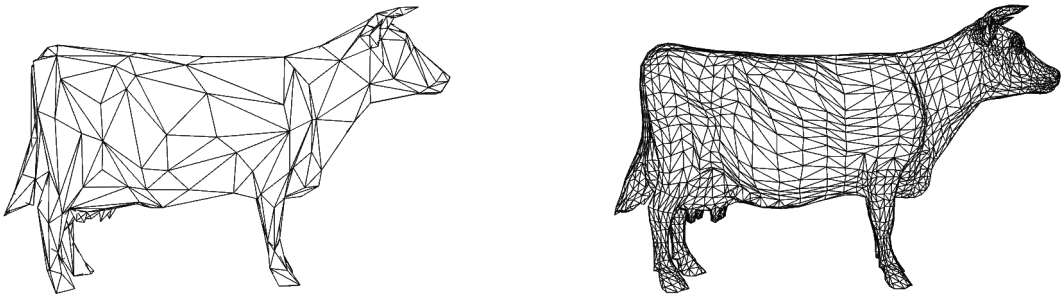

Figure 1: Simplification of the cow model to one tenth of the number of the vertices

through collapsing would be at an angle higher than $\pi/4$ then do not allow the collapse. *Hint:* Make sure the mesh is unchanged after you return from this function.

5.3 Priority of a halfedge

Compute the priority of a halfedge and return it in the function `priority()`. Review the list of modifications presented in section 4.0 to know how to compute the error introduced by an eventual contraction of this vertex.

5.4 Simplification main loop

Implement the main loop of the `decimate()` function. In every iteration there are three steps to be done.

1. Remove the vertex with the lowest priority from the queue.
2. In case the corresponding collapse is legal collapse the vertex as described in 4.0.
3. Update the queue vertices whose adjacent triangles has been changed.

Your final result should look like Figure 1 for the simplification to 1/10-th of the vertex number.

Hint: Note that the `enqueue_vertex()` functions works for both vertices which are already in the queue and completely new ones.)