CS 523: Computer Graphics
Assignment 2: Selection GUI + Local Mesh Operations

Handout: Tuesday, February 15 — Due date: March 22, 2011

1 Implementation

In this assignment you will extend your mesh viewer from assignment 1 with some face/edge/vertex selection tools, and implement a few local mesh operations. Your compilable code must fulfill the following requirements

1. The application must contain (boolean) selection of faces, edges and vertices using rectangle, lasso and single click selection tools. Element selection (faces, edges or vertices) as well as the selection tool (rectangle, lasso or single click) must be selectable from the UI (e.g. with radio buttons). Boolean selection operations must include *union* and *subtraction*. Note that rectangle selection is simply a special case of lasso selection, if you decide to tessellate the area enclosed by the sketched lasso.
2. The application must be able to extend/shrink the selected set of faces, edges or vertices by repeated dilation/erosion. The atomic structuring element for faces and vertices is given by the 1-neighbors defined by the Weiler relations. For edges, use the structuring element for faces (a center face and its three neighbors) or vertices (a center vertex and its 1-ring neighbors).
3. The selected set must be rendered in a distinct style. Selected vertices should be rendered using small spheres or circles. If a single element (face, edge or vertex) is selected, display the element's index in the UI (either on the screen, or in a separate display pane, but not in the console window).
4. Using OpenMesh's local operations, your application should be able to collapse or flip a single clicked edge of the mesh. Furthermore, your application should implement Laplacian mesh smoothing to the selected set of vertices by applying iterated local vertex relocation operations (i.e. by moving the center vertex along the graph or cotangent Laplacian vector by some scalar fraction $s \in [0, 1]$).
5. For a selected (connected) set of vertices, compute the PCA axes and visualize the results as coordinate axes emanating from the centroid, and a tight (semi-transparent) bounding box with clearly visible edges.
6. (Bonus) Using the PCA axes, fit a bivariate quadratic polynomial to the selected points and visualize it semi-transparently. This will require sampling the polynomial for triangle mesh creation.

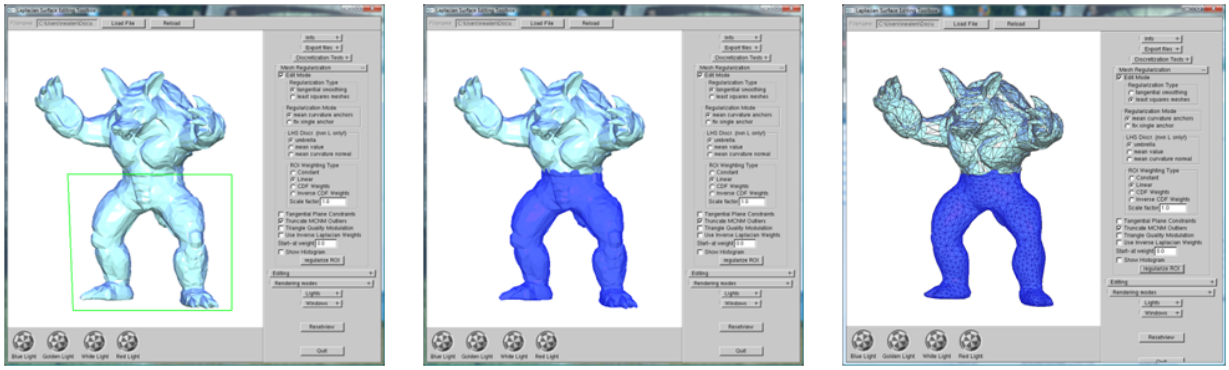

Figure 1: Selection GUI + Local Smoothing

2 Theory

1. Suppose you have a set of points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ that were sampled from some real-world data. For example, these could be points coming from some sensor, collected in a physical experiment, statistical data or data coming from a 3D scanner. You are a scientist that wants to perform principal component analysis on the data you collected and to find a line/hyperplane that approximates your points. However, it is known that the dataset contains *noise* and *outliers* – some points might be far off (see Figure 2). How do errors in the samples affect the result of the PCA? (Hint: outliers will confuse/divert the PCA. Explain why, based on the quadratic expression that PCA minimizes.)
2. Try to suggest ways to overcome the problem in the previous question.
3. One of the everyday applications of PCA in computer graphics is computation of tight bounding boxes for complex objects, as mentioned in class. However, in general, PCA does not give the minimal-volume bounding box. Explain why. (Hint: think how PCA would be affected by varying density of the points on the object, for example, if in some part of the object there is a very high concentration of points.)

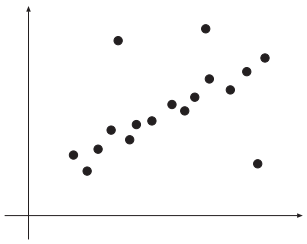

Figure 2: The points represent the measured data. There is noise and some points erroneously fall far off (they are *outliers*).