Technical tips: Step 3 (continued)

1. Computing volumes

Computing the volume of a mesh is, in principle, simple: Consider a triangle t_i of this mesh with vertices ($\mathbf{x}_1$,$\mathbf{x}_2$,$\mathbf{x}_3$). Consider a separate reference point, e.g. the mesh barycentre **o**. Then, the volume enclosed by the mesh is given by

$$V = \frac{1}{6}\left|\sum_{t_i}(v_1 \times v_2)\cdot v_3\right|$$

where $\mathbf{v}_1 = \mathbf{x}_1 - \mathbf{o}$, $v_2 = \mathbf{x}_2 - \mathbf{o}$, and $\mathbf{v}_3 = \mathbf{x}_3 - \mathbf{o}$.

This formula works also for meshes with concavities or with *tunnels* (a doughnut is a mesh with a tunnel in the middle; don't confuse this with *holes* which are discussed next).

There are however two problems in applying this formula directly:

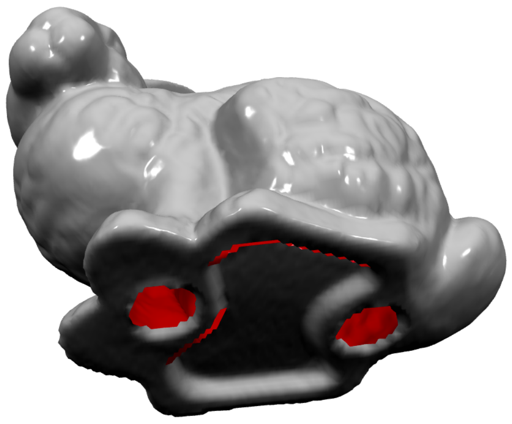
(1) Consistently oriented meshes

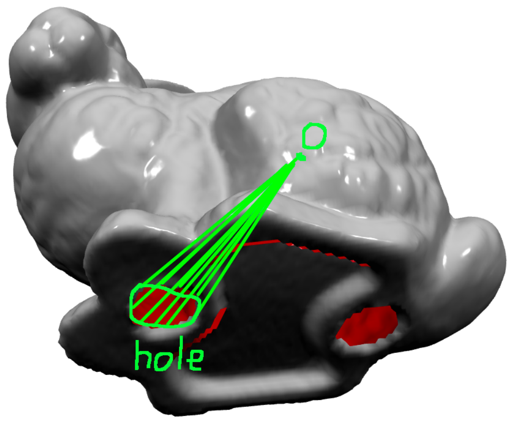
For the above volume estimation to work, the triangles of the mesh must all be consistently oriented, i.e. with normals pointing all outwards or inwards. If not, then some of the tetrahedra volumes may be *subtracted* when they need to be added, leading to a final *smaller* volume than expected. See the technical tips step 3 (previous document) for ways to check and fix triangle orientations.

(2) Meshes without holes

A hole in a mesh means that there are triangles whose edges are not shared by *exactly two* triangles. More precisely, such edges are part of a single triangle. We call these *boundary* edges. A set of connected boundary edges forms a *hole* in the mesh. Intuitively, a mesh with holes is not watertight: If we were to fill it with water, the water would run out through one or more holes.

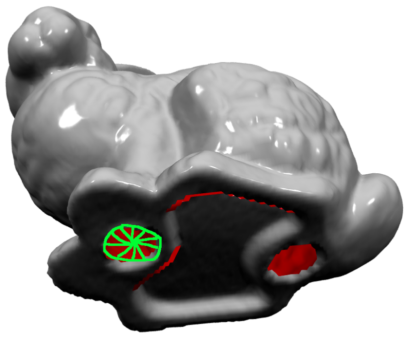
If we have meshes with holes, and estimate their volume by the above formula, the obtained volume will be *smaller* than the volume of the mesh where the hole would be stitched. More precisely, we would miss the volume of a cone-like shape whose base is the hole boundary and the apex is the point **o**. See the images below:





How can we fix such problems?

- **Detect:** First, check if there are holes. Iterate over all triangles in the mesh. List their edges and check if each edge appears in exactly one other triangle. Edges which appear in only one triangle form the boundaries of one or more holes.
- **Stitch holes:** If there are holes, find their boundaries. For this, search for edges (in the above-detected set) which connect to each other (like ($\mathbf{v}_1$, $\mathbf{v}_2$)($\mathbf{v}_2$, $\mathbf{v}_3$)…($\mathbf{v}_n$, $\mathbf{v}_1$)). The set of vertices ($\mathbf{v}_1$, $\mathbf{v}_2$,…, $\mathbf{v}_n$) forms then the closed boundary of a hole. There can be several such holes in a mesh. For each such hole, compute the barycentre **c** of its vertices vi. Then, add to the mesh the triangle fan formed by triangles having **c** as one vertex and ($\mathbf{v}_i$, $\mathbf{v}_{i+1}$) in the edge-set of the hole boundary as the other two vertices. See the image below:

  Note that this works well only for relatively planar and close-to-convex holes. Also, for this to work, you need to have the vertices on the hole boundary oriented consistently with the other polygons of the mesh, so that the triangles created in the fan have the same normal as the others in the mesh.
- **Use hole-stitching tools:** There are many such tools for repairing defects in meshes in packages such as Trimesh, MeshLab, or PMP. They are far better than the simple heuristic outlined above, especially for large and non-planar holes. However, they need a bit more reading and experimentation (and possibly coding) to get started with.

2. Checking feature extraction

In Step 3, you will compute numerous descriptors, e.g. area, volume, compactness, A3, D1, …., D4. How to check that their computation went well? There are several steps to doing this:

- Observe that there are essentially two kinds of descriptors: *Scalar* ones, represented by a single value (e.g. area, perimeter, compactness); and *distribution* ones, represented by a histogram (A3, D1, …, D4).
- Pick, manually, a few (2..3) shapes which you know that they should be very different from the perspective of each descriptor. For example, considering volume, pick a skinny shape (pencil, chair, table) and a fat shape (bunny, ball, car). Then, show the shapes and their respective descriptor values. You should see if the descriptors agree with what you visually see. Note that their *absolute* values are not important, but their *relative* ones (ratios). For instance, if a ball appears to be 10 times 'fatter' than a pencil, then its volume should be roughly 10 times larger than the one of the pencil. Note that *normalization* is essential here: The above comparison can only be made if the two shapes are scaled to the same bounding-box (e.g. unit-sized cube).
- For some descriptors, it is possible to assess their actual *absolute* values. For example, eccentricity is really how much longer than thicker a shape is, something that you can visually check quite easily.
- Histogram descriptors are pretty abstract by definition. Hence, here you cannot easily relate them to actual properties you *see* in a shape. You can check them however based on the assumption that they should deliver similar values for similar shapes and different values for different shapes. To test this: pick 3 shapes, two of a very similar kind (e.g. in the same class), and one of a very different kind (in a different class). Show the shapes and their respective histograms. The first two histograms should be quite similar, and both quite different from the third one.