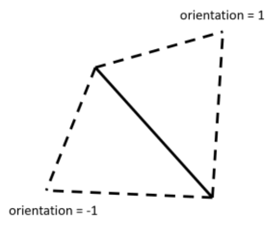
Unfolding

test/test_unfolding.py: tests used during development

src/Unfolding.py: an example script for using the functions in utils_unfolding.py

src/utils_unfolding.py:

- **create_simplified_tessellation(label, num_vertices=30)** creates a simplified tessellation of the masked region in label using *pymeshlab* with a target number of final vertices = num_vertices
    - Inputs:
        - label: 3D numpy array label map (binary values)
        - num_vertices: number of vertices (default = 30)
    - Outputs:
        - verts: vertices of the tessellation
        - faces: faces of the tessellation
- **unfold_tessellation(verts, faces, base_triangle, draw)** unfolds the tessellation created by *create_simplified_tessellation* into 2D
    - Inputs:
        - verts, faces: 3D vertex coordinates and triangle faces from create_simplified_tessellation
        - base_triangle: index of the first triangle to draw (this will be the middle of the unfolded image)
        - draw: if ==1, this function will also plot the triangles of the unfolded tessellation
    - Outputs:
        - verts_2d: 2D vertex coordinates
        - faces_2d: 2D faces
        - dict_2d_3d: dictionary that relates the index of a vertex in 2D to a vertex in 3D (multiple 2D vertices can point to the same 3D vertex due to the nature of the unfolding)
    - Additional helper functions called by *unfold_tessellation*:
        - **draw_2d_triangle(vertices)** draws a triangle defined by *vertices*
        - **find_2d_coordinates(vertices_3D, vertices_2D, orientation)** finds the 2D coordinates of the third vertex of a triangle of 3D coordinates *vertices_3D* when the 2D coordinates of the first 2 vertices are in *vertices_2D*. *orientation* determines which "side" of the 2 given vertices to put the third one on:



- **unfolded_layers(verts, faces, verts_2d, faces_2d, dict_2d_3d, im, n_layers)** exports the layers parallel with the surface of the tessellation
    - Inputs:
        - verts, faces: 3D vertice coordinates and triangle faces from *create_simplified_tessellation*
        - verts_2d, faces_2d, dict_2d_3d: 2D vertice coordinates and triangle faces, and the correspondence dictionary between 3D and 2D vertices from *unfold_tessellation*
        - im: 3D numpy array containing the grayscale image
        - n_layers: number of layers to be exported on both sides of the tessellation
    - Outputs:
        - layers: 3D numpy array where each slice contains a layer parallel to the surface of the tessellation. It contains 2*num_layers+1 layers in total and the middle slice contains the values directly on the surface of the tessellation
    - Additional helper functions called by *unfolded_layers*:
        - **get_perp_layers(coord_3d, coord_2d, im, n_layers)** extracts all layers from *im* for a single triangle in the tessellation represented by *coord_3d*, and *coord_2d* coordinates in 3D and 2D respectively
        - **triangle_area(vertices)** calculates the area of a triangle with coordinates in *vertices*
        - **rotate_im_and_mask(im, mask, angle, axes)** rotates both *im* and *mask* by *angle* about axes *axes*