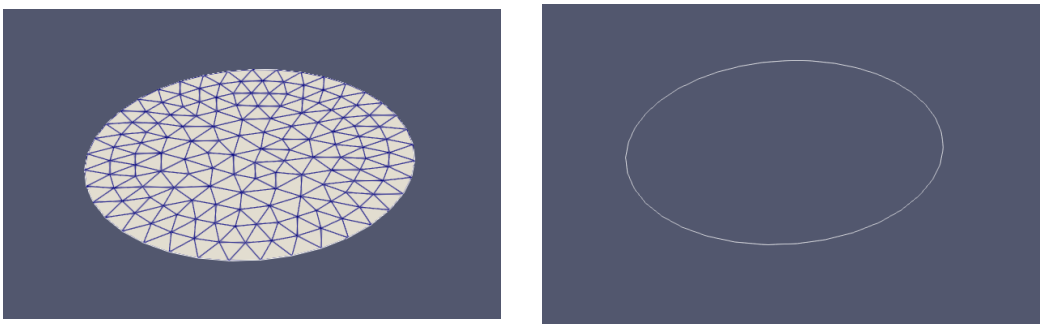

(a) Before (b) After

Figure 1: Example of skinning a mesh composed of 2D facets.

1 Building

2 Using

2.1 Using the examples

There are three examples built in to the repo. These examples utilise the functionality found in the library to different extents, and should be enough for most users to do what they need.

2.1.1 skinner

The first example, the source of which can be found in problems/skinner.cpp, is an example of how to get the skin of a mesh. This example loads in an input mesh given by the user and utilises the function `getSurfaceMesh` to obtain the skin of the input mesh. This can be used with meshes composed of 2D or 3D elements. In the 2D case, the skinned mesh will be composed of 1D edge facets, representing the boundary of the 2D boundary. In the 3D case, the skinned mesh will be composed of 2D facets.

For example, if you had a mesh called ‘myMesh.e‘, from the command

line you could enter

```
./build/examples/skinner -i Meshes/myMesh.e
```

to skin a mesh called myMesh. This should successfully save a mesh called MyMesh$_skin.einthedirectorywherethecommandwascalledfrom.$

2.2 Using the library

As previously mentioned, the functionality found in this repo is built into a library called "VacuumMeshing". The example executables are linked to this library in order to utilise the funtionality. There is no reason that this library couldn't be linked to any of your own projects. Within the repo, naviating to "./VacuumMeshing/CMakeLists.txt" will show you how the library is built. The libIGL libraries that are linked in when the library is created have PUBLIC "inheretence". This means that any targets depending on this library will automatically be linked in with the necessary libIGL libs. This saves the user (you) having to mess around with ligIGL includes/libs. The MPI lib linked in with the library is set as PRIVATE, so that when building your own projects they will NOT link against the MPI version used to build the VacuumMeshing library.