Documentation

2D Mixed–element mesh generator

Version 2018.11.23 ROI–type

Claudio Lobos
clobos@inf.utfsm.cl
Departmento de Informática – UTFSM – Chile.
Fabrice Jaillet
fabrice.jaillet@univ-lyon1.fr
Université Lyon 1 – LIRIS – France.

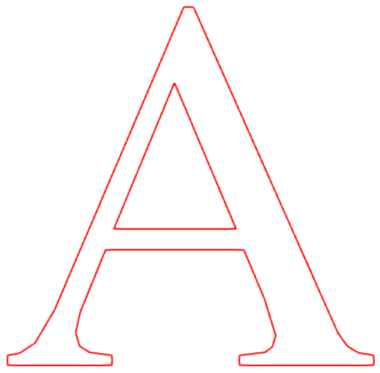
UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Abstract

This document will show you how to obtain mixed–elements meshes with the provided code. Two main alternatives are explained here. The first will show you how to use the code as a standalone program. The second will show you how to bundle your own code with the mixed-element mesh generator. In both cases, mixed–element are employed to manage transitions between fine and coarse regions and at the boundary of the domain. All the remaining regions will be meshed with structured regular quadrangles.

1 Data Structure

Quadrant, QuadEdge, Polyline
Final resulting Mesh



(a)

Figure 1: An example of complex polyline with holes and sharp features. This corresponds to the Input to be meshed.

2 A quadtree-based approach

mixed-elements mesh generation, describe here the main steps method

2.1 a quadtree-based method

show different steps of the method:

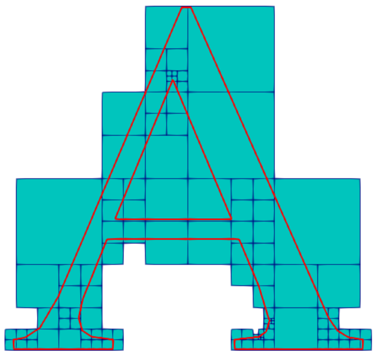
Algorithm 1: Generation process

```
Result: A mixed-elements mesh
1 foreach Quadrant do
      repeat
2         Subdivide Quadrant;
          foreach new Sub-Quadrant do
              if Intersects Input or Is Inside then
                  Insert Sub-Quadrant
              end
          end
      until desired Refinement Level;
  end
3 Create balanced quadtree;
4 Apply Transition Patterns;
```

1. subdivision, quad + ROI + off-domain quad removing
2. balanced : the resulting mesh is not balanced if one of the edges of a Quadrant is subdivided twice. In this case, subdivide the Quadrant as in Algorithm 1 step 2. And repeat until the mesh is balanced.
3. transition patterns (OMP)

2.2 Boundary handling

preprocess



(a)

Figure 2: After boundary handling.

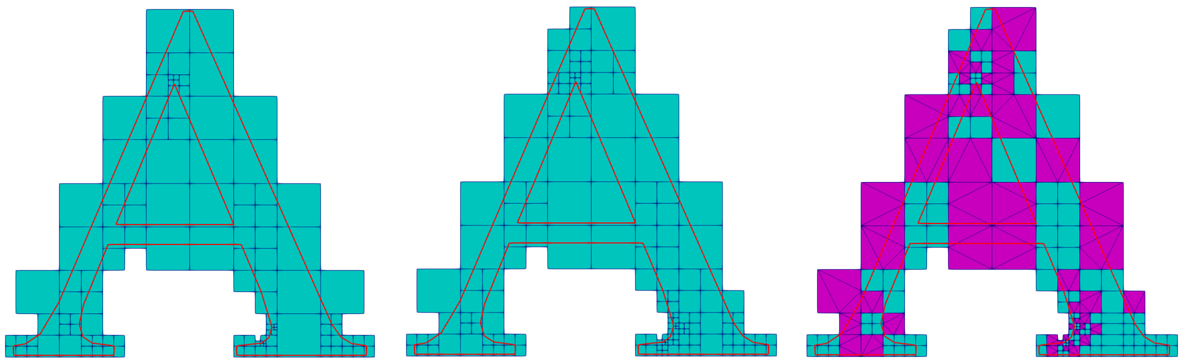

(a) (b) (c)

Figure 3: After refinement, balancing and transition steps, level 3.

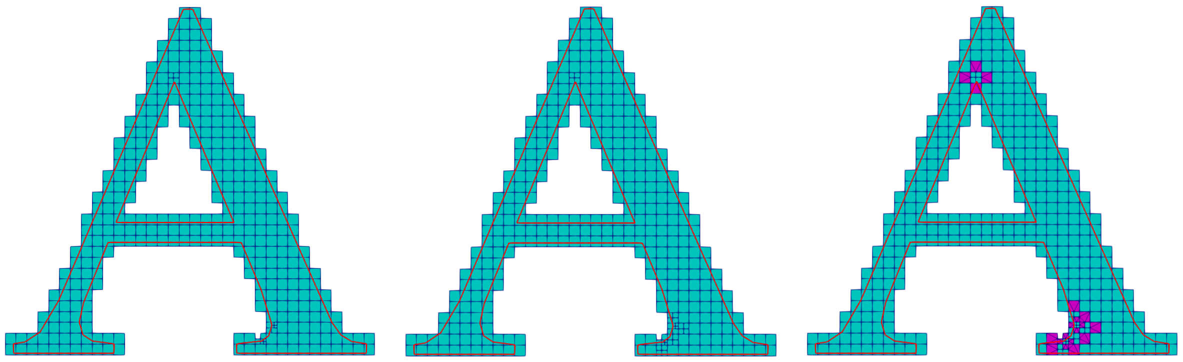

(a) (b) (c)

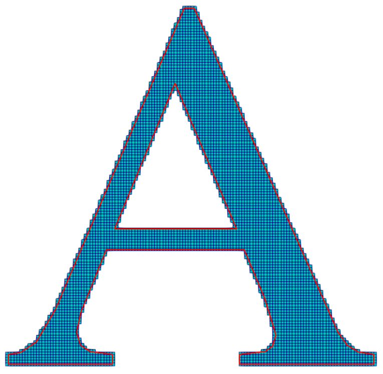
Figure 4: After refinement, balancing and transition steps, level 5.

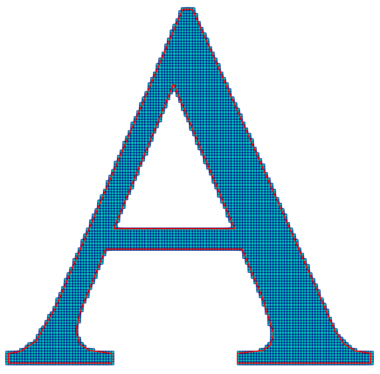
2.3 Refinement, Balancing and Transition steps

3 Fitting quadrants to Input Surface

Algorithm 2: Generation process and Input surface fitting

```
  Result: A mixed-elements mesh
1 foreach Quadrant do
      repeat
2         Subdivide Quadrant;
          foreach new Sub-Quadrant do
              if Intersects Input or Is Inside then
                  Insert Sub-Quadrant
              end
          end
      until desired Refinement Level;
  end
3 Create balanced quadtree;
4 Apply Transition Patterns;
5 Detect Features in Input;
  Project Close to Boundary;
  Remove on Input Surface;
  Shrink to Boundary;
6 Apply Surface Patterns;
```


(a)


(b)

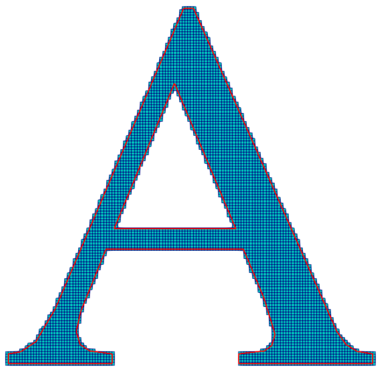
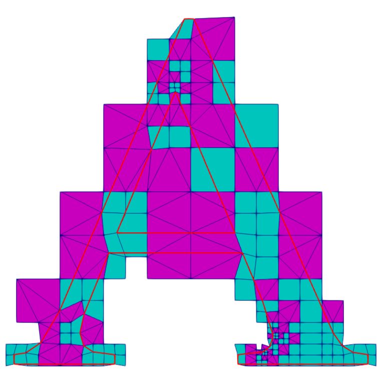
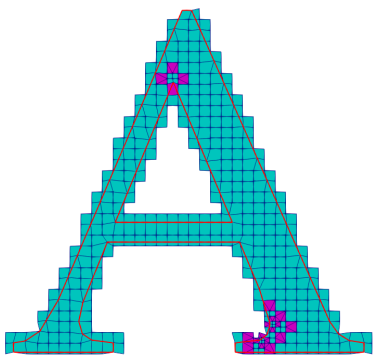

(c)

Figure 5: After refinement, balancing and transition steps, level 7.

3.1 Project interior points close to boundary


(a)


(b)

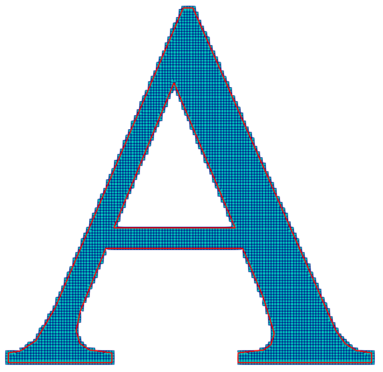

(c)

Figure 6: After projecting interior points close to boundary, level 3, 5, 7.

3.2 Remove on input surface

3.3 Surface Patterns

surface patterns + features

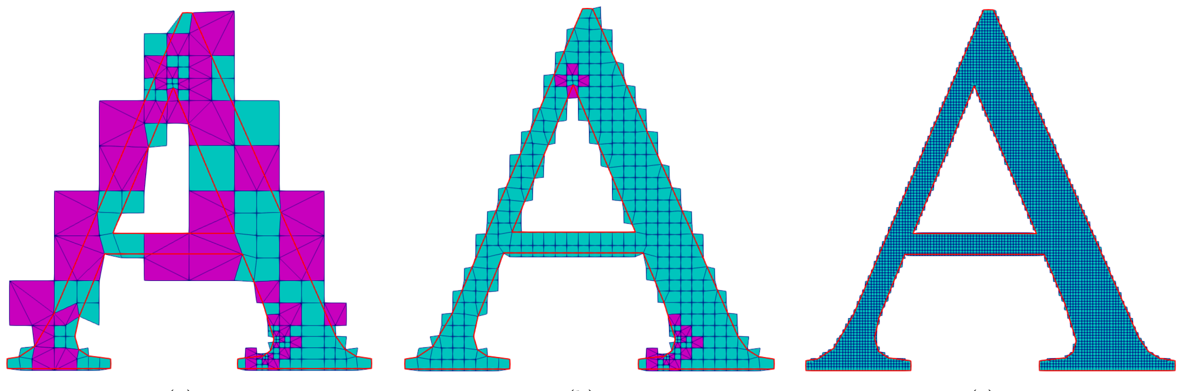

(a) (b) (c)

Figure 7: After safely removing exterior quadrants, level 3, 5, 7.

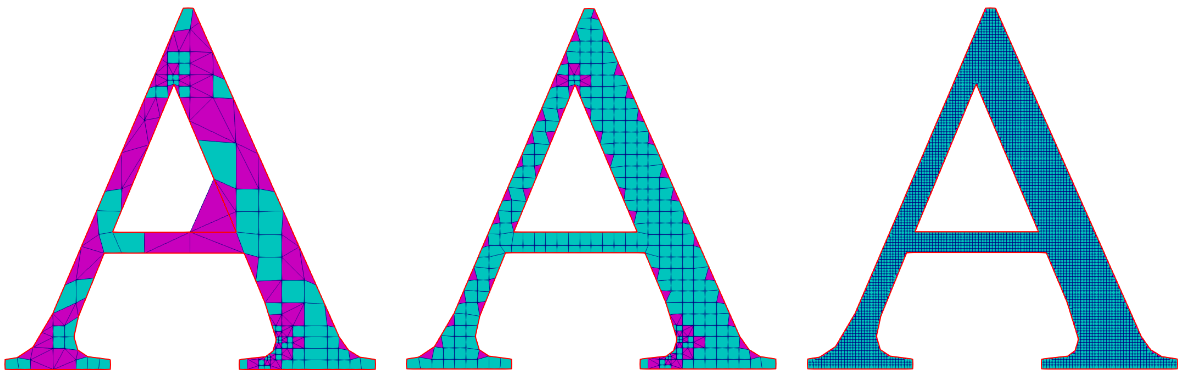

(a) (b) (c)

Figure 8: After applying surface patterns, level 3, 5, 7.

4 Remeshing

Algorithm 3: Refinement process

```
Result: A refined mixed-elements mesh
foreach Element to refine do
    Identify containing Quadrant;
    Subdivide Quadrant;
    foreach Sub-quadrant do
        if Intersect input or Is In then
            Insert Sub-Quadrant
        end
    end
end
Goto Algorithm 1, step 3
```

identify which Quad contains the element, information from file
subdivide Quad + goto subsection balanced
(local remeshing? neighbourhood??)

5 Installing

In order to install this application you will need a Unix–based system, a `c++` compiler and `cmake`. You should do the following:

```
$ unzip mesher.zip
$ cmake src/
$ make
```

This has been tested on Linux and Mac without any error nor warning.

Acknowledgement

Several researchers have contributed with ideas, guidance, and general discussion to this project. Special thanks to Nancy Hitschfeld, Yohan Payan, Marek Bucki, Vincent Luboz.
Also several great students have contributed in coding and integrating with other software. Many thanks to Eugenio González, Sebastián Durán, Esteban Daines, Cristopher Arenas and Sebastián Tobar. This list will continue to grow because we are currently implementing new features that will add great value to this meshing tool and their will be available in a future version.
This work was financed by grant: Fondecyt de Iniciación 11121601 and Ecos-Conicyt C11-E01 and C16-E05.