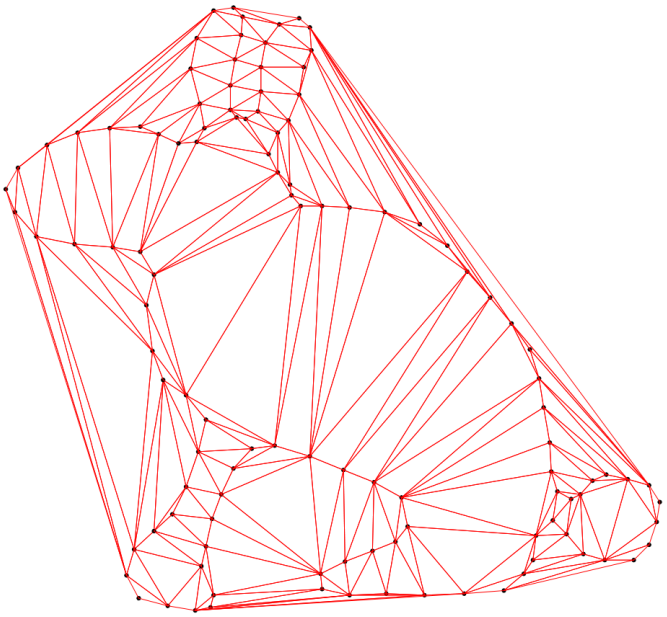
$3\mathcal{D}$ $\mathcal{M}$odelling

Assignment 3; Contour reconstruction

S.J. van den Elzen (F072754)
M.K. Lewiński (F072790)



Universiteit Utrecht

Game and Media Technology

March 24, 2009

Contents

1 Introduction

The third assignment for the course *3D Modelling* at Utrecht University stated the implementation of a 2D version of the algorithm described by Boissonnat [Boi84]. In this document the used method, the implementation and conclusion are described. The method describes the 2D version of the algorithm. The implementation gives a short description of the implementation. Detailed compile instructions can be found in appendix A.

2 Method

Roughly the two-dimensional Boissonnat method for contour reconstruction is as follows: Given a pointset P. First the delaunay triangulation and convex hull of P are calculated. Then the currentContour is initialized to the convex hull of P. Then it is checked which triangles of the delaunay triangulation have 2 vertices which are neighbours on the currentContour. Then for each of these triangles the distance from the edge (with points on the currentContour) to the circumscribed circle are calculated. Then the triangles are sorted with descending distances. The triangle with the greates distance is then removed and the currentContour is updated. This process repeats until currentContour does not change anymore.

Algorithm 1 $2DBoissonnat(S)$

Require: A 2D pointset S – $(x, y) \in \mathbb{R}$.
Ensure: The reconstructed contour of S.

```
1:  P ← convexhull(S)
2:  D ← delaunay-triangulation(S)
3:  currentContour ← P
4:  while currentContour changes do
5:    for each triangle t in D do
6:      if 2 points (p1, p2) of t are neighbours on currentContour then
7:        calculate distance d from (p1, p2) to circumscribed circle
8:        add t to T
9:      end if
10:   end for
11:   sort T according to descending distance d
12:   remove first(T)
13:   update currentContour
14: end while
```

3 Implementation

The implementation of the algorithm is done using C++ and Qt 4.4.3. For computing the delaunay-triangulation and the convex hull of the pointset, libraries are used. These libraries (*Convex Hull*, *GEOMPACK*) are both open source and can freely be used under the GNU LGPL license. The computation of the distance d from the line (with two points on the current contour) to the circumscribed circle is implemented in the following way: Say we have a triangle t with points d, e, f. Now d and e both lay on the current contour. Then

first the circumcenter c of t is calculated. The radius r of the circumscribed circle is then the distance from any point on t to c. Then the midpoint of the line d, e is calculated by:

$$midpoint = (\frac{d.x + e.x}{2}, \frac{d.y + e.y}{2}) \quad (1)$$

The distance h from the midpoint to c is then calculated and it is checked whether c lies within the current contour or outside the current contour. The final distance is then calculated according to this test:

$$d = \begin{cases} r - h & \text{,if c lays within currentContour} \\ r + h & \text{,if c lays outside currentContour} \end{cases} \quad (2)$$

4 Conclusion

The two-dimensional version of the algorithm of boissonnat is succesfully implemented. This algorithm is a fast and generic way to solve the problem of contour reconstruction. For three of the four input files our implemented algorithm gave the best result possible (as can be seen in Figure 1). For the second input file (*the bunny – prac3-2.txt*) there is a slight error near the ears. This error is probably caused by the fact that the points near the ears aren't densely enough sampled. This speculation is confirmed by some experiments. By adding just 2 points to the left ear of the bunny, the contour is perfectly reconstructed without any errors (see figure 1e). The file with the 2 added points is added to the input files *prac3-2b.txt*. To have a succesfully reconstructed contour the points should be sampled dense enough, therefor this method will not always work in practice. But nevertheless it gives good results and is reasonable fast.

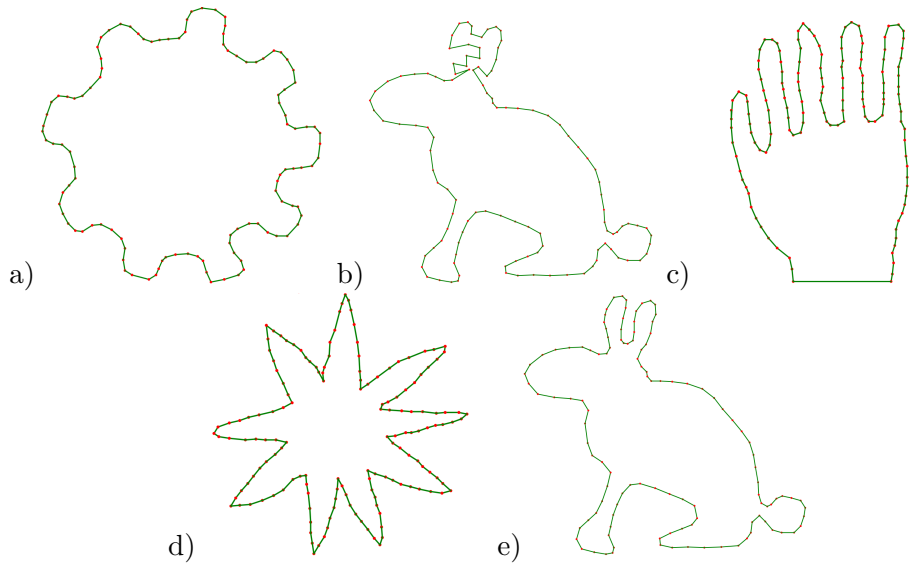


Figure 1: Results of input files a)*prac3-1.txt*, b)*prac3-2.txt*, c) *prac3-3.txt*, d)*prac3-4.txt* and e)*prac3-2b.txt*.

A Compile instructions

We chose to make the program compile with the following tools:

- **G**NU **C**ompiler **C**ollection (GCC) 4.3.2
  http://gcc.gnu.org/
- Qt 4.4.3
  http://trolltech.com/products/qt

Qt is a cross-platform application framework for C++. The main choice to work with this setup, is to provide cross-platform compilation. With this setup we can easily compile the constructed code to run on Linux, Mac and Windows. Another great advantage of Qt is that it has an extensive library with graphical user interface components which can easily be adapted to serve our needs.

This section first explains how to build and install the requirements for the `Boissonnat contour reconstruction` program, which is Qt 4.4.3. After that, the build procedure for the Boissonnat contour reconstruction program itself is given.

In this section it is assumed that the code is build for Linux. The build procedures for Windows and Mac OS X are almost the same, but will not be discussed here. The main differences for compiling on the different platforms is how to include and link the external libraries.

On each platform the program can easily be build without adjustments by loading the project file into Qt creator (free downloadable from http://trolltech.com/developer/qt-creator) and pressing the *build-all* button. At code level, platform dependent differences are taken into account (for example mouse handling) but will not be discussed here.

A.1 Qt 4.4.3

Qt 4.4.3 can be downloaded at ftp://ftp.trolltech.com/qt/source. The file needed is called "qt-x11-opensource-src-4.4.3.tar.gz". Save this file to a known directory, say "/qt". After the file has been downloaded, open up a "terminal" and run the following code (please note that for the last command you need to enter your password):

```
1 user@host:~$ cd /qt
2 user@host:/qt$ tar -zxvf qt-x11-opensource-src-4.4.3.tar.gz
3 user@host:/qt$ cd qt-x11-opensource-src-4.4.3
4 user@host:/qt/qt-x11-opensource-src-4.4.3$ ./configure --prefix=/usr
5 user@host:/qt/qt-x11-opensource-src-4.4.3$ make
6 user@host:/qt/qt-x11-opensource-src-4.4.3$ sudo make install
```

After this Qt 4.4.3 is installed. This can be verified by running "qmake -v", which outputs something like:

```
1 user@host:~$ qmake -v
2 QMake version 2.01a
3 Using Qt version 4.4.3 in /usr
```

A.2 Boissonnat

After Qt 4.4.3 is installed, The Boissonnat contour reconstruction program can be compiled. Assuming the source code of the program is placed in "/src", run the following commands in a "terminal":

```
user@host:~$ cd /src
user@host:/src qmake boissonnat.pro -config release
user@host:/src$ make
```

Boissonnat contour reconstruction is now compiled. To run Boissonnat, you can run the executable found in the directory where the source of Boissonnat is by typing in a "terminal" (from the "/src" directory):

```
user@host:/src$ ./boissonnat
```

References

[Boi84] Jean-Daniel Boissonnat. Geometric structures for three-dimensional shape representation, 1984.