Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: michael.lienhardt@laposte.net
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-91 24th September 2004

Morphological operations on surfaces

Michael Lienhardt

Abstract

In this report, we propose an implementation of some basic morpohological operations on triangulations of surfaces, in particular :

1. dilation
2. complement
3. intersection, union, subtraction of two objects on the surface.

The definition of the distance and an object on the surface are the main contributions of this work. Some other functions for the visualisation of the scene are also described.

1 Introduction

In [3], a new application domain for the implementation of the mathematical morphology theory is introduced. The idea was to take a surface in 3D space, with an object on it, and to define the morphological operation on such scenes. The purpose of this work is the implementation of these functions. In this application, the surface is represented by a triangular mesh, which gives us a graph. Our work is then greatly inspired by the already defined morphology on graphs [1], but it differs owing to the fact that the operations we want to define here are based on a distance, not on a structuring element. Another constraint of our implementation is that the distance was not specified. It could be either Euclidean, Isophotic as presented in [3], or another distance. This distance function can directly give the distance between two neighbours of the graph, but not between two unspecified graph nodes, because we cannot know if this distance will respect the shape of the surface.

In Section 2, we describe the different solutions we found to properly calculate the distance between two graph nodes. In the third section, we introduce the different functions we have implemented.

2 Definition of the distance

The morphological operations in this report are based on distance. Hence the definition of the distance is very important. As the graph gives us only the distance between two neighbouring nodes, we must find a way to approximate the distance between two nodes which can be anywhere on the graph (Figure 1).

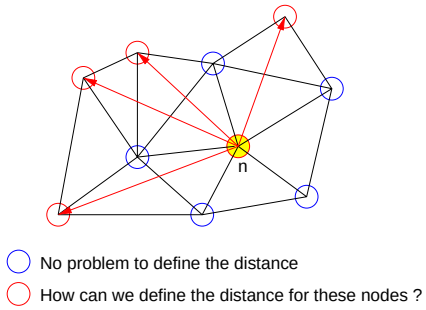

Figure 1: The problem to define the distance between the yellow node and the others

Let $G = (V, E)$ be the graph of the nodes, $n \in V$ and for all $v \in V$, we want to calculate the distance between nodes n and v, which we write as $d(v)$.

We have found three ways to define the distance $d(v)$ for all nodes $v \in V$:

1. **The minimum path** :

   We can easily compute the minimum path between two nodes in a graph.

   This way to define the distance is usual in graph theory, but here, as we can see in Figure 2, the errors are huge, and we cannot consider this solution as a good one.

   Actually, the difficulty to define the distance is caused by the graph being in a 3D space. And so, we cannot apply the Euclidian distance between two nodes, because this method would not respect the curve given by the graph. But there are methods to resolve this problem and to compute the distance in a 2D space. For instance, the previous method computes the distance in a 1D space (the two neighbours draw a line). But we have not enough information in 1D space, so we must consider not only a line (two neighbours), but a plane (three neighbours : a triangle).

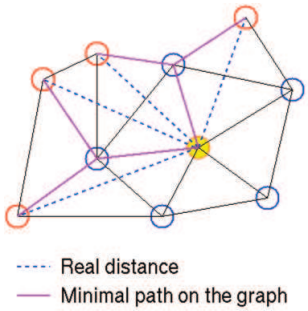

Figure 2: The first method using the minimum path

2. **The triangle solution** :

Let us consider that we have two nodes v and w which are neighbours, and that we know $d(v)$ and $d(w)$ (from previous computations). We want to calculate the distance of a third node u, neighbour of the two previous nodes (see Figure 3).

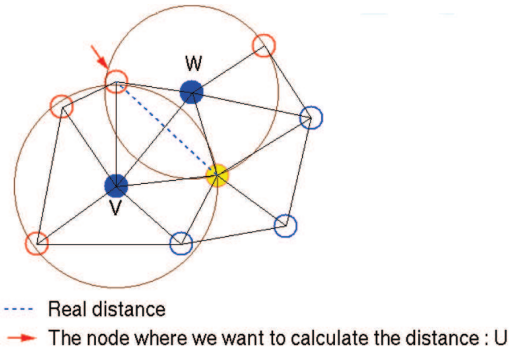

Figure 3: The second method : reconstruct the center of the distance field

The idea of this solution is to reconstruct in the 2D space, with the distance information, two triangles : (v, w, u) and (n, v, w). As we can easily find out, the method doesn't give any error if the surface given by the graph is plane, and we can suppose that the results are good on whatever curve we use.

But first of all, we must know how to construct a triangle in 2D space in knowing only the distance between the nodes.

Problem : reconstruction of a triangle from the distance information

Data : the distance between the three nodes : $d(n_1, n_3) = a$, $d(n_2, n_3) = b$ and $d(n_1, n_2) = c$.

Result : A triangle with the good distance properties in 2D space

Solution : (*using the notation of Figure 4*) We put the first node on $(0, 0)$, the second one on $(0, c)$. For the third node, we have to calculate v_1 and d, and we put it at (d, v_1). We have

$$\left\{ \begin{array}{rcl} c & = & v_1 + v_2 \\ a^2 & = & d^2 + v_1^2 \\ b^2 & = & d^2 + v_2^2 \end{array} \right. \Longrightarrow \left\{ \begin{array}{rcl} a^2 - b^2 & = & v_1^2 - (c - v_1)^2 \\ \sqrt{a^2 - v_1^2} & = & d \end{array} \right.$$

And then, $v_1 = \frac{c^2 + a^2 - b^2}{2c}$.

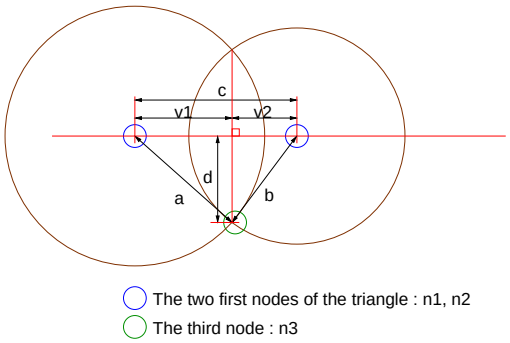

Figure 4: The method to reconstruct a triangle from distance information

We just need then to construct the nodes u and n with the algorithm we have just find out, and the distance is $\|u - n\|$

The only problem with this solution is that the distance we want to define is not always from a node of the graph to another, but also from an object to a node, and sometimes, we cannot construct the second triangle, as shown in Figure 5.

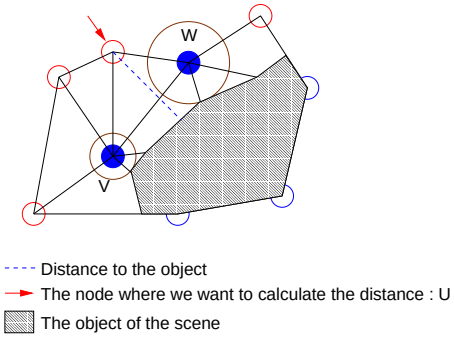

Figure 5: The fourth node doesn't exist : the two circles don't intersect themselves

So, for this case, we have to find another solution.

3. **<u>The border line solution</u>** :

The idea of this solution is to reconstruct in the 2D space the object border line, instead of constructing a node. Let us consider, as in the previous method, that we know the distance for the nodes v and w, and that we want to calculate the distance of u, a neighbour of the two previous nodes.

As for the previous method, we reconstruct in the 2D plane the triangle (v, w, u), And then, we must have a method to construct the line.

Problem : reconstruction of a line with the distance information of two nodes

Data : the distance between the two nodes and between the nodes and the line : $d(n_1, n_2) = dist_3$, $d(n_1, l) = d_1$ and $d(n_2, l) = d_2$.

Result : A line and two nodes with the good distance properties in 2D space. The line is defined with two of its points.

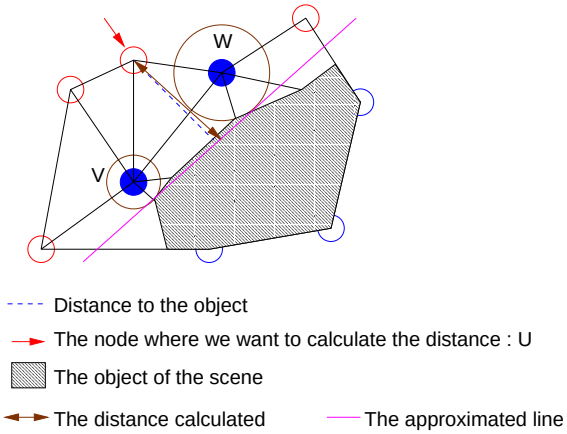

Figure 6: The third method : reconstruct the object border line

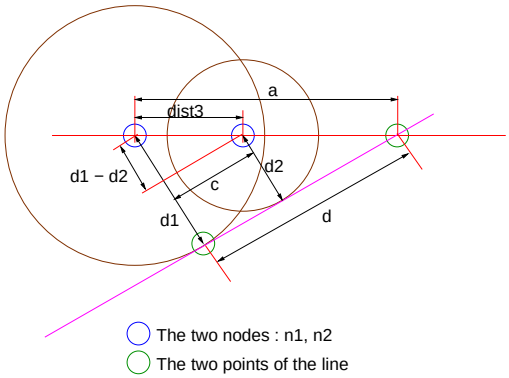

Figure 7: The method to reconstruct a line from distance information

Solution : (*using the notation of Figure 7*) We put the first node on $(0, 0)$, the second one on $(0, dist3)$. If $d_1 = d_2$, then the two points of the line are $(d_1, 0)$ and $(d_1, dist3)$.
We can now suppose that $d_1 > d_2$. On Figure 7, a can be computed with Thalès's theorem : $a = \frac{dist_3 d_1}{d_1 - d_2}$. So we have the first point of the line : $(0, a)$. For the second point, we can use the previous method to construct a triangle : we only need the distance d. And $d = \frac{c.d_1}{d_1 - d_2}$ with $c = \sqrt{dist_3^2 - (d_1 - d_2)^2}$

This method works in every condition, but causes some errors. In Figure 8, we can see that if we want to know the distance from a node, then the computed distance is not the good one, and even when the distance is from an object, we have an error because the object border line is not a line, but a set of segments (Figure 6).

3 The implemented functions

In this implementation, an object is known as shown in Figure 9 : we store the distance information for each graph node in an array called *distance*. The distance for each node on both sides of the object border line are the distance to the line. It is very important for the dilation and the definition of the line, as presented in the further sections.

This information is contained in a bigger structure which stores all the information about the scene. The structure is divided into two parts : the graph part, and the object part, which are described next :

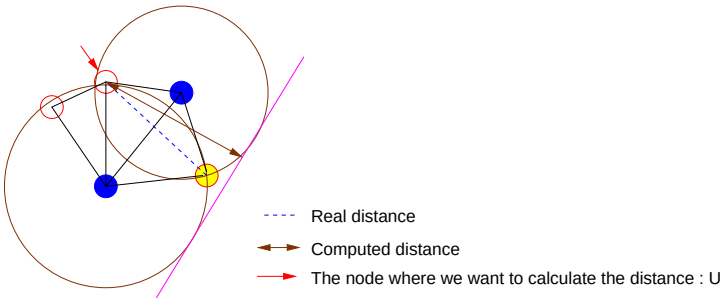

Figure 8: One error of the third method

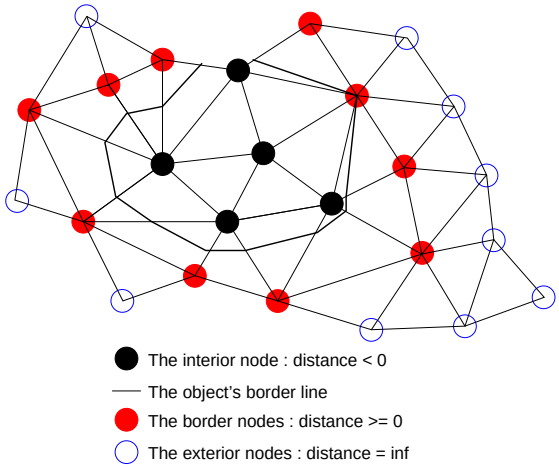

Figure 9: The implementation of the object information

graph part

This information is stored in an array *dim* of three structures, one cell for each dimension (the nodes, the edges, the triangles).

- *node* : For each node, here is its coordinates, its neighbours in the graph and the edges for which it is an extremity.
- *edges* : here is stored the length of each edge, and also the two nodes and two triangles (or one if the edge is on the border of the scene) neighbours of the edges.
- *triangles* : We only need to know which nodes and which edges constitute each triangle. And also the area of each triangle to compute the color.

object part

- *distance* : the array previously described.
- *dFOBL* : it is the array of the distance from the border of the object for each node
- *borderLine* : here are stored the nodes where the object border line cuts the edges of the graph. This information is a result of the *distance* information, so it is only computed when it is necessary.
- *color* : the color of the triangles. As for the previous information, it is computed only on the demand.

With this implementation, we only have to use the *distance* (and the *graph*) information to compute all the morphological operations. So, it is very powerful, but still, it has one problem shown in Figure 10. As only one distance is stored for each node, we cannot reconstruct the initial object shown in the image. This problem is very difficult to avoid, so we have considered that it was a good compromise.

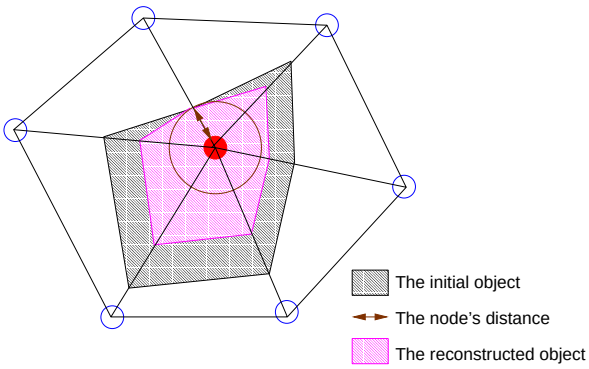

Figure 10: The problem of this implementation : we cannot have many distances for one node

In the following sections, we keep the $G = (V, E)$ notation and add one:

$$v, w \in V, \quad v \mathcal{N} w \iff \text{the two nodes are neighbours}$$

3.1 The dilation

The idea of the algorithm is to simulate the growing of the object, as shown in Figure 11 : the border of the object moves towards the object's exterior, as the distance parameter allows it.

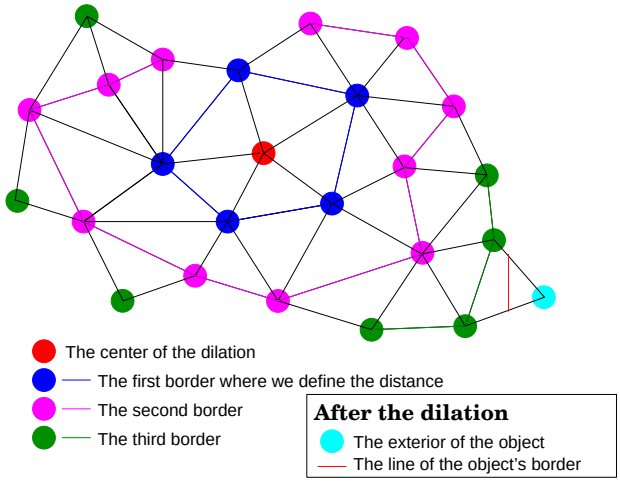

Figure 11: The principe of the dilation

This method allows the use of the different methods to define the node's distance : for each border, we can calculate the distance of its nodes from the distance of the previous border's nodes. And the next border is made up of the exterior neighbours of the border's node where the distance is smaller than the distance parameter d.

border $\leftarrow \{v \in V \mid 0 \leq distance(v) < d\}$: the border nodes, which will be part of the dilated object.

while($border \neq \emptyset$)

$research \leftarrow \{v \in V \mid \exists e \in border,\ e\mathcal{N}v\ \&\ distance(v) = \infty\}$: the exterior neighbours of the border nodes

calculate the distance for each node of the set *research*.

$border \leftarrow \{v \in research \mid distance(v) < d\}$

endWhile

for each $v \in V$, $d(v) \leftarrow d(v) - d$ endFor : the object grows

Actually, this method encounters some problems. A first problem is shown in Figure 12 : some nodes of the current border have only one neighbour belonging to the previous border, and then, we cannot use either the second nor the third distance method on these nodes, using as reference points only the node of the previous border. In this case, we must use also the nodes of the current border, where the distance is already defined.

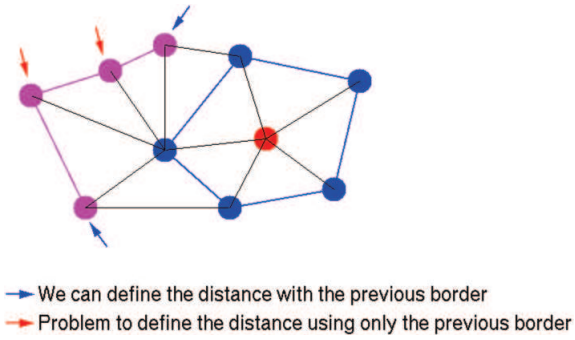

Figure 12: The problem of the 1 connected node

The second problem is presented in Figure 13. It occurs when we have many centers of dilation, and the border of two centers meet themselves. The indicated node has not a good distance because it was reached from a further node than the nearest one. In this case, we must recompute the distance of the node.

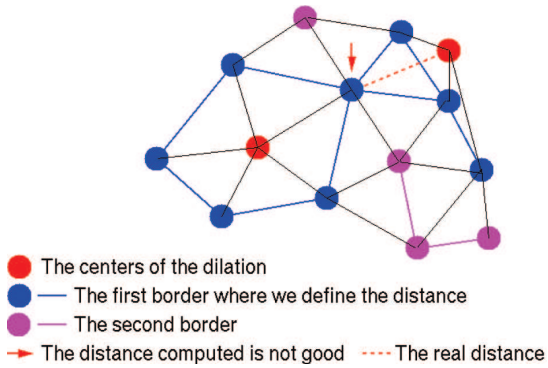

Figure 13: The problem of two borders which meet themselves

All these problems put together, we can no longer consider only the exterior nodes to construct the next border, and not only the previous border to define the distance. We propose then the following algorithm, which constructs the next border with all the nodes of the graph which have a neighbour within the border, i.e. with a distance smaller than that of the border (we haven't put the initialisation lines, the 'while loop', nor the last line of the algorithm presented in the introduction of this part) :

$research \leftarrow \emptyset$

for each $b \in border$

for each $u \in \{v \in V \mid v\mathcal{N}b \ \&\ d(v) > d(b)\}$

$dTmp \leftarrow \min(d(u), d(b) + d(u,b))$: the minimum path method

for each $w \in \{v \in V \mid v\mathcal{N}b \ \&\ v\mathcal{N}u \ \&\ d(v) < \infty\}$: all the triangle where a distance is defined

$dTmp \leftarrow \min(dTmp(u), \underline{distThirdMethod}(b,u,w))$

endFor

if($dTmp < d(u)$)

$d(u) \leftarrow dTmp$

if($d(u) < d$) $research \leftarrow research \cup \{u\}$ endIf

endIf

endFor

endFor

$border \leftarrow research$

By this algorithm, we use the first and the third distance method, and it resolves all the presented problems. However this solution is not as good as it could be, because the algorithm calculates many times the distance for the same node, and for some nodes we don't need to know the real distance (the interior one). Another problem is that we never use the second distance method which can give better results in some cases.

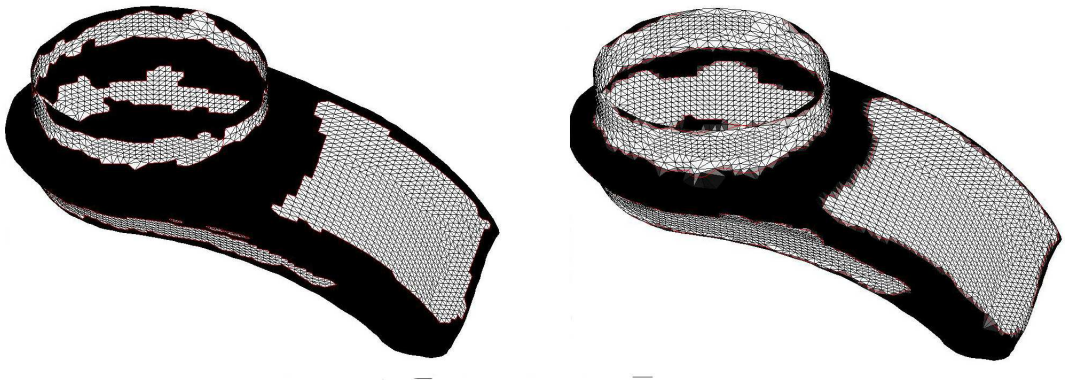

Figure 14: An example of dilation : the right image is the dilated with Euclidian distance of the left image (the display uses the function described in the visualisation section)

3.2 The skeleton

A first way to compute a skeleton is given by VINCENT [4]. He proposes to compute the distance from the object's border to all the interior nodes, then the nodes with a local maximal distance constitute the skeleton. The computation of the interior nodes' distances is nearly the same algorithm as for the dilation. The local maxima are then very simple to find.

But this method doesn't keep the homotopy of the object. The idea to restablish the connectivity between the different local maxima is to follow the crest line of the same distance information as required for VINCENT's algorithm. A crest line reconstruction algorithm is also proposed by Meyer [2]. The algorithm we propose to construct the crest line is based on two ideas :

1. The detection of the crest's minima

2. The method to climb the crest from these nodes.

Let $c \in V$ be a local minimum of the crest line, and S the set of its neighbours. We can divide S into three parts :

- $S_1 = \{v \in S \mid d(v) < d(c)\}$
- $S_2 = \{v \in S \mid d(v) > d(c)\}$
- $S_3 = \{v \in S \mid d(v) = d(c)\}$

As c is a node of the crest line, S_1 is never empty, and as c is a minimum, neither is S_2. The property of c is that the sub-graph generated by S_1 (resp. S_2) is disconected (Figure 15).

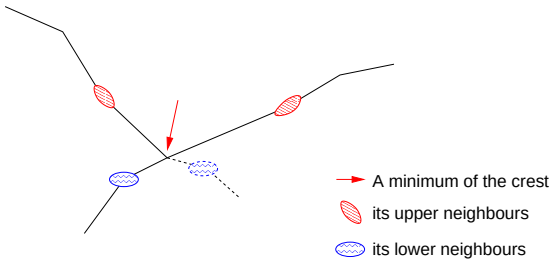

Figure 15: The properties of the minima of the crest

Let p be a node of the crest line and S the set of its neighbours. To climb the crest line from p, as previously, we divided S into three parts. The next nodes are the local maxima of the sub-graph generated by $S \cup \{p\}$ (Figure 16).

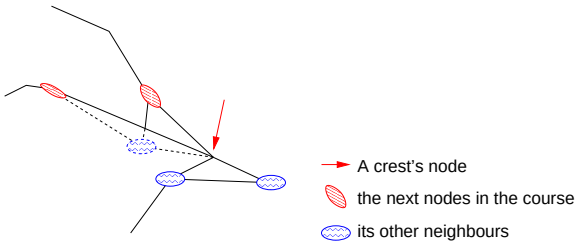

Figure 16: The course of the crest

This method works well in the theory, but as we work in a non-continious space, the detection of the local minima suffer some problems in some configuration of the graph.

3.3 Functions on sets

As mathematical morphology is based on set theory, it was interesting to implement some set functions, such as the union, intersection, and complement. The objects are known by the node's distance to the object. Hence two objects give two distances for each node of the scene. Hence a union is just to keep the minimum distance for each node, and the intersection, the maximum distance.

The calculation of the complemented object is slightly more complicated.

> *distance* $\leftarrow -$*distance* : the opposite distance to have the opposite object. And then, we have to change the distance of the exterior node : it must be ∞.
>
> for all $v \in \{v \in V \mid d(v) > 0\}$

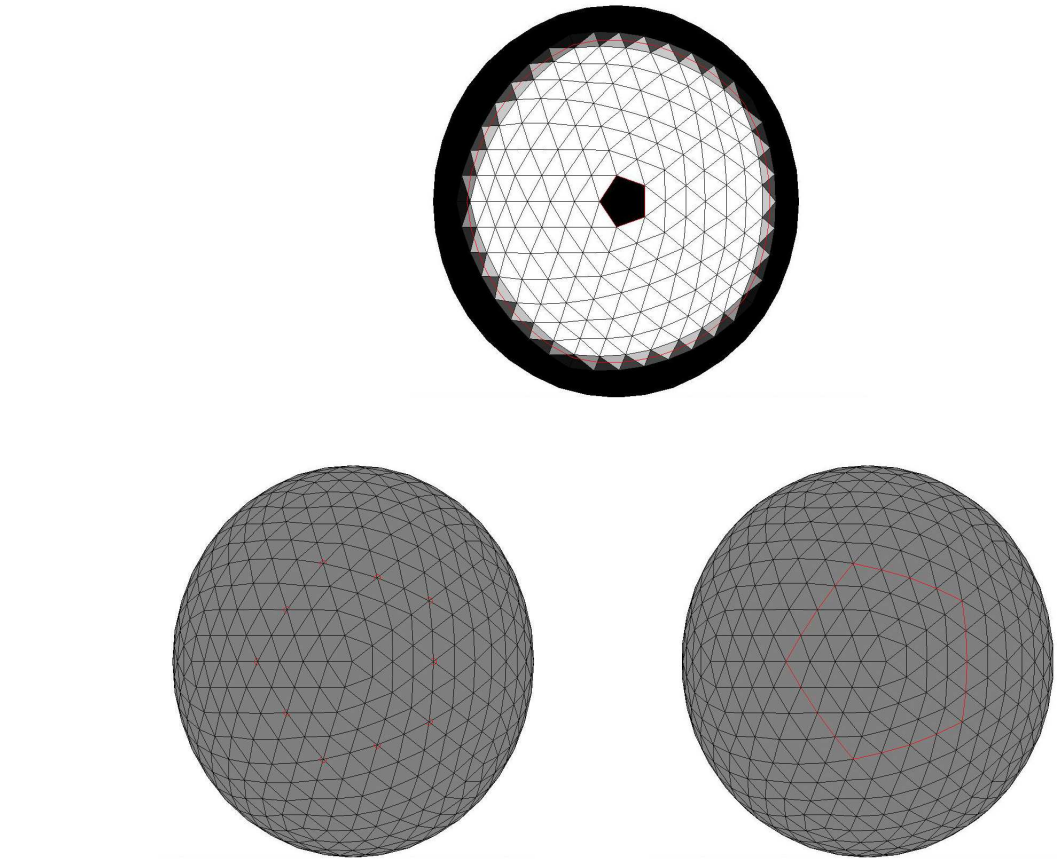

Figure 17: Two examples of skeleton for the upper image : The one with the VINCENT's algorithm, and the homotopic skeleton

$S \leftarrow \{w \in V \mid w \mathcal{N} v\}$
if($\min_{w \in S}(d(w)) \geq 0$), $d(w) \leftarrow \infty$ endIf

endFor

3.4 Visualisation

The implementation of the object, in keeping only for each node of the graph its distance to the object, is very useful for the previous functions. But this information is not really concerning the object and for the visualisation, we need to know where it really is.

3.4.1 The border of the object

This function calculates the line of the object's border. It is divided into two parts : first, we compute the nodes of the line, and then the line itself in calculating the neighbours for each line's node.

In the first part of the function, we compute two things :

1. the coordinate of each node of the border line.
2. the relation $\mathcal{R}$ between the nodes of the border line and the edges of the graph :

$$\forall n \in borderLine,\ e \in E,\ n \mathcal{R} e \iff n \text{ is on the edge } e$$

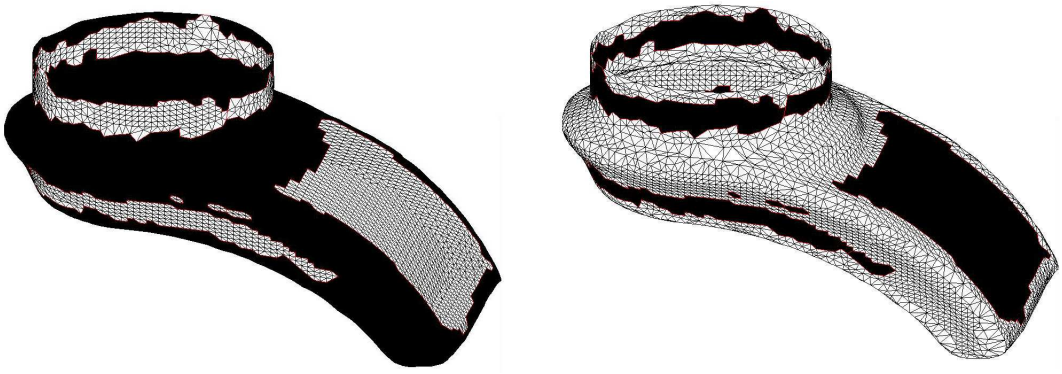

Figure 18: An object, and its complement

We need this information for the computation of the neighbour of the node on the border line, and for the definition of the triangle's color.

As we can see in Figure 20, most of the line's nodes have only one or two neighbours, and stand only on one edge. But the nodes with a distance 0 from the object are on a node of the graph, and then are on many edges and can have more than two neighbours when it connects different parts of the object. In the following algorithm, we compute these nodes apart from the others, because of their properties.

$borderLine \leftarrow \emptyset$

for each $b \in \{v \in V \mid d(v) > 0 \ \& \ d(v) < \infty\}$: the strict border nodes.

for each $i \in \{v \in V \mid i\mathcal{N}b \ \& \ d(i) < 0\}$: the interior neighbours of b.

$dist \leftarrow \frac{d(b)}{d(b)-d(i)}$

$node \leftarrow node(b) + dist(node(i) - node(b))$: the coordinate of the node.

push($borderLine$, ($node$, (b, i))) : the coordinates and the related egde.

endFor

endFor

for each $b \in \{v \in V \mid d(v) = 0\}$: the graph nodes which are in the border line

push($borderLine$, ($node(b)$, $upNeighbour(b)$)) : the coordinates and the related edges.

endFor

The second part of the function uses the property that two neighbours on the border line are on the same triangle, as shown in Figure 21.

The idea to compute the line is to take, for each node, its neighbour triangles. Then the neighbours are the object line's nodes which are on these triangles. As we haven't access directly to the triangle neighbours, we use the relation $\mathcal{R}$ previously defined :

for each $n \in borderLine$

$triNeighbour \leftarrow upNeighbour(e \in \{e \in E \mid n\mathcal{R}e\})$

$edges \leftarrow downNeighbour(triNeighbour)$

$neighbour \leftarrow \{m \in borderNode \mid \exists e \in edge, \ m\mathcal{R}e\} \setminus \{n\}$

endFor

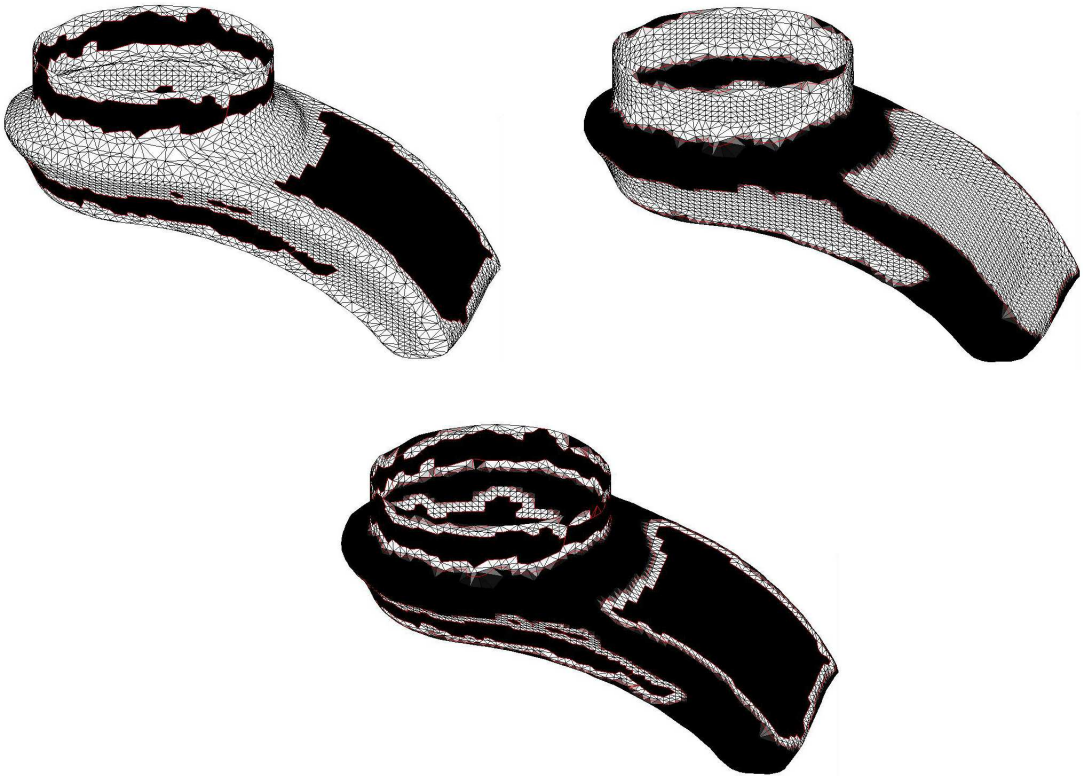

Figure 19: Two objects and their intersection

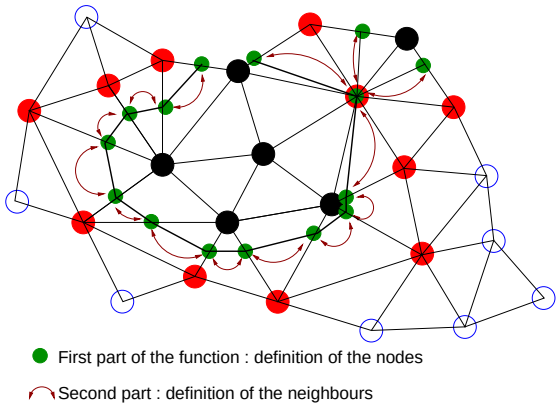

Figure 20: The principle of the function

3.4.2 The color of the triangle

The most usual example of the implementation of the mathematical morphology theory is the black-and-white/binary images. There, the basic element is the pixel, and the object is all the white pixels. Hence, a basic element is either in the object, or in its complement, and then it is very simple to draw the object of the scene. In our case, the basic element is the triangle, and an object can cover only a part of it. So, to visualise the object, we no longer consider a binary image, but triangles with grey scale color : a value ranging between 0 and 1. For each triangle, its color is the information 'How much the object covers the triangle' : $\frac{area(triangle \cap object)}{area(triangle)}$

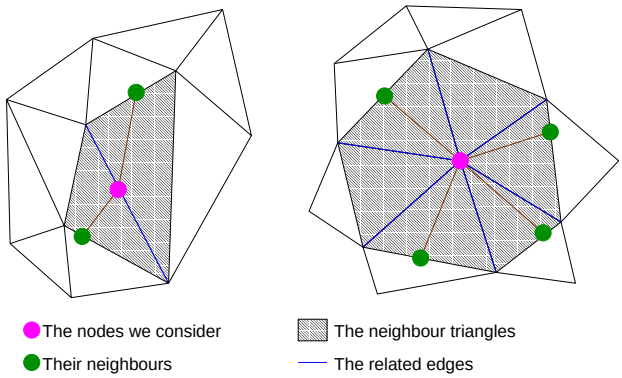

Figure 21: The relation between the node neighbours and the triangle neighbours

The algorithm to compute the color is relatively simple : for each triangle, we detect if it is entirely covered by the object, if it is exterior to the object, or if it is cut. Only in this last case, do we compute the previous formula. There is one case when we cannot compute the color of one triangle : when all its nodes are at a distance 0 from the object : this one is in or out the triangle, we cannot know. So, in this case, we keep the previous color for the triangle.

for each $t \in triangles$

$interiorNode \leftarrow \{v \in V \mid d(v) < 0\} \cap node(t)$

$borderNode \leftarrow \{v \in V \mid d(v) = 0\} \cap node(t)$

$exteriorNode \leftarrow \{v \in V \mid d(v) > 0\} \cap node(t)$

if$(\sharp(interiorNode \cup borderNode) = 3)$, $color \leftarrow 1$: the triangle is white

elseif$(\sharp(interiorNode) \neq 0)$: the triangle is cuted

$lineNode \leftarrow \{n \in borderLine \mid \exists e \in downNeigbour(t),\ n\mathcal{R}e\}$

$partObject \leftarrow interiorNode \cup lineNode$: the nodes of the polygon $t \cap object$

$color \leftarrow area(node(t))/area(partObject)$

elseif$(\sharp(borderNode) \neq 3))$, $color \leftarrow 0$

endIf

endFor

4 Conclusion

In this report, we have presented a way to implement some basic morphological operations on triangulated surfaces, which are based on distance. These operations are the base of the mathematical morphology theory, and by composing them, we can obtain the majority of the morphological operations. It is a first step in a research domain which can lead to many results in image processing and recognition. However this topic needs more research than described here, because many points are not deepened enough, such as the dilation algorithm and the distance. For instance, the dilation function should include the second distance method, and could calculate in less time all the needed distances.

References

[1] H. J. A. M. Heijmans, P. Nacken, A. Toet, and L. Vincent. Graph morphology. *Journal of Visual Communication and Image Representation*, 3(1):24–38, 1992.

[2] F. Meyer. Skeletons and perceptual graphs. *Signal Processing*, 16(4):335–363, 1989.

[3] Helmut Pottmann, Tibor Steiner, Michael Hofer, Christoph Haider, and Allan Hanbury. The isophotic metric and its application to feature sensitive morphology on surfaces. In *Proceedings of the ECCV 2004 Conference*, volume 4, pages 561–572, 2004.

[4] Luc Vincent. Graphs and mathematical morphology. *Signal Processing*, 16:365–388, 1989.