Istituto di Matematica Applicata e Tecnologie Informatiche

Consiglio Nazionale delle Ricerche

Genova – ITALY

SHREC

Version 1.0

SHape characterization and structuring through **RE**eb graph **C**omputation

Silvia Biasotti

USER MANUAL

Index

1 Introduction 3
1.1 PLATFORM REQUIREMENTS 3
1.2 INPUT Format 3
1.3 OUTPUT Format 3
1.4 USER REFERENCES: 4
2 GUI functionalities 4
2.1 Choice of the measuring function: 4
2.1.1 Height function: 5
2.1.2 Barycentre distance: 5
2.1.3 Integral geodesic distance: 5
2.1.4 Geodesic distance: 6
2.1.5 Curvature extrema distance: 6
2.1.6 User's function: 6
2.2 Insertion of contour levels 6
2.2.1 NSections: 6
2.2.2 Insert constraints: 6
2.2.3 Vis. Rings: 7
2.3 Surface characterization 7
2.3.1 Characterize: 7
2.4 Filtering and local refining of the surface characterization 7
2.4.1 Filter Max/Min: 7
2.4.2 Filter Saddles: 8
2.4.3 Hole Contouring: 8
2.5 Critical area expansion and Reeb graph visualization 8
2.5.1 Expansion: 8
2.5.2 Vis. Graph: 8
2.5.3 HIDE TIN: 8
2.5.4 Vis. Skeleton: 8
2.6 Output storage 8
2.6.1 Save Attributed graph: 9
2.6.2 Save graph.iv: 10
2.6.3 Save Segmentation: 10
2.6.4 Save Skeleton .top/.geo: 10
2.6.5 Save Skeleton VRML 2.0: 11
2.6.6 Save TIN: 11
3 TUTORIAL 11
4 REFERENCES: 17

1 Introduction

SHREC is a tool for characterizing (in the sense of computing maxima, minima and saddles) a surface without boundary with respect to a real function by computing a set of contour levels. Starting from such a characterization, SHREC is also able to compute a Reeb graph of the surface. The tool proposed here is an implementation of the algorithms and the measuring functions described in [1][2][3]. SHREC is freeware for non commercial purposes.

1.1 PLATFORM REQUIREMENTS

Linux, with operating system RedHat 7.2 at most, OpenInventor and Motif libraries.
The tool is stand-alone. The library is written in standard C language, while the interface uses OpenInventor.

1.2 INPUT Format

A connected, regular triangle mesh without boundary coded in the ".ver", ".tri" format.

The ".ver" file must contain the # of vertices of the triangulation (TIN) followed by the list of coordinates (x y z).
The ".tri" file must contain the # of triangles and the list of faces. Each triangle is coded through six indices: the first three indices correspond to the "position" of the triangle vertices in the ".ver" file; the others are the indices of the three triangles that are adjacent to the current one.
File converters from VRML 1.0 and .off formats to the ".ver", ".tri" format (and vice versa) are provided.

Example of file ".ver", ".tri": a planar triangle mesh of 4 triangles and 6 vertices.
File ".ver":

```
6
0 0 1
4 0 1
2 2 1
4 4 1
6 2 1
8 0 1
```

File ".tri":

```
4
1 2 3 2 0 0
3 2 4 1 3 0
4 2 5 2 4 0
5 2 6 5 0 0
```

Each "zero" in the indices of the file ".tri" denote that that triangle has an edge on the mesh boundary.

1.3 OUTPUT Format

Several kinds of output are possible.
In the user's terminal: a list of critical areas, nodes and coordinates of the Reeb graph.

Moreover, it is possible to save the Reeb graph in different formats: as a list of nodes and edges with a list of associated attributes, in an OpenInventor Scene "graph.iv", or like a skeleton: in a two text files ("skel.top" and "skel.geo") and in a VRML 2.0 scene. Moreover, it is possible to save the modifications done by the user in the TIN model in a new triangle mesh, in ".ver", ".tri" format.

Finally, it is possible to save the segmentation of the triangle mesh in significant regions (maxima, minima, etc.), the file format will be a triangle mesh in an internal format similar to the standard .off format but with a label associated to each triangle (face).

1.4 USER REFERENCES:

The executable starts from a Linux terminal with the following syntax:
./reeb nameTIN or ***./reeb nameTIN.ver***

Then the TIN model is loaded and a GUI is opened.
The TIN is displayed in a canvax, at the left of the GUI; the user's interaction is possible operating in the buttons of the two columns, at right, near to the canvax.

The user is expected to perform the surface characterization and the Reeb graph extraction according to the following pipeline:

1) to choose a measuring function;
2) to choose a number of contour levels that will be computed and displayed on the model;
3) to characterize the surface (according to the behaviour of the contour levels on the surface);
4) (optional) to eventually filter the surface characterization and to eventually refine the contour levels so that there are not holes through the surface hidden in a region;
5) to topologically expand and the critical areas (until all surface triangles are visited) and to compute the Reeb graph;
6) to display and eventually save the Reeb graph, the "centreline skeleton" and the surface segmentation.

Buttons in the two columns allow the user to interact with the GUI.

The middle column of buttons contains a list a button for normalizing the triangle mesh and a set of choices for the measuring function. To help the user during the whole process, at each step of the process only the buttons that the user can select in the GUI are enabled.

2 GUI functionalities

<NormalizeTIN>: The normalization of the model is optional. The user can (geometrically) normalize his model in any step of the process. With the normalization option, the model is translated and scaled so that the normalized model is embedded in a sphere of unitary radius, centred in the origin of the Cartesian axes.

2.1 Choice of the measuring function:

All measuring functions must be scalar, that is they must associate to each vertex of the TIN only a real value. The user can choose his favourite measuring function in the list of the functions proposed or upload a new function through a file.
In particular, the tool contains the following measuring functions:

- ✓ the height function, functions <*Height function*> and <*Rotate TIN*>; <*Rotx*>, <*Roty*> and <*Rotz*>;
- ✓ the distance from the centre of mass, function <*Barycentre distance*>;
- ✓ the distance from the centre of the smallest sphere that inscribes the model, function <*BoundingSphere distance*>;
- ✓ the geodesic distance from a seed point [5] lying on the TIN, function <*Geodesic distance*>;
- ✓ the integral geodesic distance proposed in [4], function <*Integral geodesic distance*>;
- ✓ the minimum geodesic distance from several seed points (generally, vertices having high curvature, according to the "Tailor" tool [5][7]), function <*Curvature extrema distance*>;
- ✓ a user defined function, function <*User's function*>.

When a measuring function has been selected, the TIN vertices are coloured according to the value of the function in them: the redder, the higher the value is; the bluer, the lower the value of the function is.
The user may repeat the choice of the measuring function but, when the system continues the characterization process, only the last user's choice will be considered.

The behaviour of each function is explained in the following.

2.1.1 Height function:

The height function associated simply associates to each TIN vertex its elevation (z-value). If the user wants to change the canonical z-direction (the z-direction of the model is parallel to the user view direction) he may rotate the object by selecting a rotation vector from the sliders <*Rotx*>, <*Roty*>, and <*Rotz*> and pushing on the button <*Rotate TIN*>. At this point, a new embedding of the model in the 3D space has been obtained and the new height function can be computed from the button <Height function>.

<Rotx>*:* Such a function allows the user to select the x-value of the rotation vector he wants to apply to the model; the x-value may vary in the interval [-1, 1] and it is 0 if no choice is done;

<Roty>*:* Analogously to <Rotx> but respect to the y-value;

<Rotz>*:* Analogously to <Rotx> but respect to the z-value;

<Rotate TIN>*:* The rotation function physically rotates the model, according to the x/y/z-values of the rotation vector ; if no choice of the vector is done (vector *(0,0,0)*) the function does not make effect.

2.1.2 Barycentre distance:

The distance from the centre of mass B associates to each vertex of the TIN its Euclidean distance from B. The centre of mass is computed in a computationally robust way that may also treat TIN with non-uniform vertex distribution; in fact, the contribution of each vertex is weighted with the area of its star and the total sum is normalized with the area of the model.

2.1.3 Integral geodesic distance:

The integral geodesic distance the button computes is an approximation of the smooth function proposed in [4]. In the implementation of the SHREC tool, a brute force approximation of the function based on the Dijstra algorithm that requires $O(n^2 \log(n))$ operations is implemented. Therefore, such a computation is time expensive and it is suggested to evaluated such a function only on

"reasonably" small TINs (for example less the 15.000 vertices). See [4] for a further and computationally cheaper approximation of such a function.

2.1.4 Geodesic distance:

The geodesic distance from a seed point on the TIN is computed through the Dijstra algorithm. Its computational cost is O(nlog(n)) and may be computed in two manners. If there is no choice of the seed point by the user, the user randomly chooses a starting point. Otherwise the user can select a point on the mesh through the standard OpenInventor pick option. Then, selecting the button "Geodesic distance", the system computes the seed point as the most distant vertices from the user selected point, see [5] for details on the criterion of choosing the seed point and the geodesic is computed.

2.1.5 Curvature extrema distance:

For the computation of the minimal geodesic distance from a set of points it is expected that there exists a file, named "***nameTIN.ikk***". Such a file should contain the number and the list of 3D coordinates of seed points.
Example of the file ".ikk":
3
1 0 0
2 1 0
3 2 1
The user may select a set of points on the model by picking the TIN vertices in the canvax with the standard OpenInventor option and save the coordinates that are written in the Linux terminal in an opportune file.

2.1.6 User's function:

Finally, the user could upload any measuring function he want. The only requirement is that the function is real and valued in each TIN vertex. To upload such a function the user must generate a file "***nameTIN.usr***" that lists the values of the function in the TIN vertices, in the same order they appear in the ".ver" file.

When the measuring function has been computed at least once, the <*Insert constraints*> button is enabled.

2.2 Insertion of contour levels

Once the measuring function has been computed the model is ready for the insertion of a set contour levels, that is the set of points in which the function is constant. At this stage of the tool on a uniform contouring of the model is allowed.

2.2.1 NSections:

The user chooses the number of level sets N. Such a number must be strictly positive.

2.2.2 Insert constraints:

Contour levels are inserted as constraints in the TIN (adding new vertices in correspondence of the intersection between contours and edges on the TIN) according to the method proposed in [1]. In fact, the number N of level sets induces the number of intervals in which the domain of the function is subdivided. For example, let us suppose the domain of the measuring function is the interval

[0,1]. Choosing "N" level sets, the user computes all contour levels that have value [1/(N+1), 2/(N+1),…,N/(N+1)], that is the interval [0,1] is divided in (N+1) sub-intervals.

2.2.3 Vis. Rings:

Such a toggle allows the user to visualize or not the contour levels.

2.3 Surface characterization

2.3.1 Characterize:

The surface characterization is performed according to the criteria proposed in [1], extracting critical areas through a region-growing process the ends when the boundary of a region is done only of constrained edges.
Given a region R and its boundary B_R, the following classification scheme is adopted:
✓ R is a *maximum* (resp. *minimum*) area iff all the outgoing directions from B_R are descending (resp ascending);
✓ R is a *saddle* area iff the number of components of B_R is greater than three and there are both ascending and descending outgoing directions from B_R;
✓ finally, R is called *regular* iff does not belong to the previous categories.
In addition to the previous classification scheme, a further distinction between simple and multi-connected minimum and maximum areas is done: *simple* critical areas are minima (resp. maxima) that correspond to a simply-connected region and *complex* the other ones.

In the canvax, regular regions are in light grey.
Red areas correspond to maxima, blue ones to minima while saddles are in green.
A summary of the region classification is displayed also in the Linux terminal.

2.4 Filtering and local refining of the surface characterization

All three functionalities described in this section (functions <*Filter Max/Min*>, <*Filter Saddles*> and <*Hole Contouring*>) are optional and can be used to adapt the surface characterization to different contexts. Once these button are selected the surface characterization will change according to the choices done and a summary of the new characterization will appear on the Linux terminal.
In fact, the adopted uniform slicing guarantees that all features having size greater than the size of the sub-intervals induced by the choice of the level sets are detected. Features whose size is less than the dimension of the interval are discarded, except those that extend across a level set.
The adopted characterization criterion is dependent on the frequency of the slicing process: if the frequency is too low, we might lose some important features, such as small holes completely contained within two adjacent contour levels. It is easy, however, to detect these situations and adapt the frequency of the feature size simply by using the Euler formula for each region.

2.4.1 Filter Max/Min:

To make the filtering effect homogeneous, the contour behaviour is re-computed so that maxima and minima that are adjacent to another critical areas but the local variation of the values of the function is less the interval size are collapsed to the

other regions. The result on the graph of such an operation is to prune leaf nodes that connect to adjacent regions. In this way all the features having size greater than the sub-interval size are recognised and the smaller ones are discarded.

2.4.2 Filter Saddles:

Another option for simplifying the surface characterization and diminishing the number of critical areas is to collapse also saddles that are adjacent. In this case the region collapse is done only if after the region merging step no holes completely belong to a same region.

2.4.3 Hole Contouring:

This option modifies the surface characterization and the contouring only if such holes exist; in that case the final subdivision of the domain of the function may be not more uniform.

2.5 Critical area expansion and Reeb graph visualization

2.5.1 Expansion:

Such a function computes the Reeb graph using an algorithm based on a region growing process that works in two steps. First, arcs between minima (resp. maxima) and saddles are inserted by connecting all maxima/minima to their nearest (in terms of region expansion) critical area; then, the graph is completed through a region growing process that stops when all possible ascending directions and arcs have been visited, details are in [1].
TIN elements are coloured according to the region they belong during the region expansion process: light red (resp. blue) triangles have been encountered during the expansion of maxima (resp. minima), while light green regions are associated to the expansion of a saddle area.
Finally a summary of the classification of nodes in the Reeb graph and its structure is printed in the Linux terminal.

2.5.2 Vis. Graph:

Such a function enables the visualization of the Reeb graph like a graph-representation that associates to each node the centre of mass of the corresponding critical area and some centroids of the contour levels to arcs.

2.5.3 HIDE TIN:

This option allows to hide the TIN model and to visualize only the Reeb graph of the skeleton.

2.5.4 Vis. Skeleton:

A skeleton-like representation of the Reeb graph may be displayed. In this case the graph is regarded as a centreline that associates to each contour level its centre of mass and the connection between nodes is done according to the e

2.6 Output storage

Moreover, there are several options for coding the Reeb graph (and its skeletal representation) in a file.

2.6.1 Save Attributed graph:

Generates a file named "***nameTIN.out***" in which the graph is stored with several node and edge attributes. Firsts, the volume and the surface area of the TIN is coded.
Then, the keyword "nodes:" is placed just before the number of nodes of the Reeb graph. The list of nodes follows. For each node some attributes are stored: a label, the 3D coordinates of the representing point x, y, z, "point"; the characterization "char"; the perimeter "lenght" and the area "area" of the corresponding critical area; the average of values of the measuring function in the critical area "quote" and, finally, the average of the Euclidean distance between the boundary of the critical are and the point "boundary".In particular, the char field of the graph may assume the following values: "0" for minima, "1" for saddles, "2" for maxima; nodes that are along an arcs and do not correspond to any critical area are characterized with "3" and their fields "length", "area", "quote" and "boundary" are zero.
The list of arcs starts after the keyword "edges:" and the number of edges. Each edge is stored with the indices of the starting and ending nodes; the length of an arc is stored in the field "steps" that represents the number of contour levels crossed during the expansion process.

An example of the file is in the following:

```
volume: 0.013662
nodes: 5
node
 label      0
 point      0.500402 0.503856 0.558525
 char       0
 lenght     0.002049
 area       0.404604
 quote      0.562500
 boundary   0.001467
node
 label      1
 point      0.502231 0.517180 0.876591
 char       2
 lenght     0.000499
 area       0.031533
 quote      0.750000
 boundary   0.000507
node
 label      2
 point      0.504154 0.725710 0.859187
 char       2
 lenght     0.000226
 area       0.007747
 quote      0.875000
 boundary   0.000237
edges: 4
edge
 0 2
 steps       5.000000
edge
 0 1
 steps       3.000000
```

2.6.2 Save graph.iv:

Saves the Reeb graph as a OpenInventor scene and generates the file: "***graph.iv***", in the same directory where the executable is.

2.6.3 Save Segmentation:

Such a function generates a file "***Segmented.off***" that codes the surface segmentation induced by the expansion of the critical areas.
To each maximum or minimum area is associated the portion of the surface identified by growing the border of the region, until the boundary of another critical area is reached. Therefore that is the biggest mesh portion topologically equivalent to the critical area. Furthermore, considering the surface portions that correspond to the arcs beginning from saddles, we observe that each of these surface sub-parts is topologically equivalent to a disk with an hole, or better, to a cylinder without its bases. Therefore, a complete surface segmentation is obtained by considering all maximum areas topologically equivalent to maxima and minima, saddle areas and the regions between saddles, for details see [3].
The output file format is a small modification of the ".off" format; an integer label is added to each face to code the region to which the triangle belongs to. In practice, the file starts with the keyword "OFF" and the number of vertices, faces and a 0. Then, the list of vertex coordinates and of the indices of faces follow. Since in this case each face is a triangle, the indices of vertices is preceded by the number "3" and, at the end of the line, there is the label of the region.
For example:

```
OFF
6 4 0

0 0 1
4 0 1
2 2 1
4 4 1
6 2 1
8 0 1
3 1 2 3 1
3 3 2 4 1
3 4 2 5 2
3 5 2 6 2
```

In addition such a function highlights the surface segmentation directly on the model; associating same colour to triangles belonging to same region.

Another way for saving the graph is to store it like a skeleton represented through two lists of vertices and edges.

2.6.4 Save Skeleton .top/.geo:

Such an operation generates the two files "***skel.top***" and "***skel.geo***" that are coded in an internal format.
The file ".geo" contains the number and the list of points of the skeleton. A node of the skeleton is associated to each region of the constrained triangle mesh. Geometric attributes are associated to each point. In particular, points are represented by 8 values: the <x, y, z> coordinates, the area of the corresponding region, the perimeters of the upper or lower contour level of the region; a flag that

is 0 if the contour is closed and the value of the function f in the region (if the node is terminal the value of the nodes assumes the maximum or the minimum of function f in the region, while it is the average otherwise).
The file ".top" contains the number and the list of edges. Analogously to the ".tri" format, each edge is coded by the indices that correspond to the position of points in the file ".geo".

For example:
File ".geo"
7

-0.037543	-0.038318	-0.139109	0.330057	0.401587	0.175075	0	1484.1543
0.051217	0.058975	0.207554	0.140405	0.232642	0.032678	0	1669.6736
-0.001988	0.000096	0.480603	0.171847	0.193531	0.155978	0	1669.6736
-0.198781	-0.223379	-0.152943	0.024408	0.028671	0.021052	0	2040.7122
0.254717	0.290499	-0.129186	0.009527	0.011336	0.006806	0	2411.7508
0.234539	0.288229	-0.140413	0.006569	0.008407	0.002935	0	2411.7508
0.242350	0.267136	-0.123535	0.049414	0.060237	0.027081	0	2411.7508

File ".top"
6

1	6
0	3
0	1
0	6
4	6
5	4

2.6.5 Save Skeleton VRML 2.0:

The skeleton is saved in the VRML 2.0 format in the file "***skel.wrl***". For instance, such an output may be visualized using the "vrmlview" tool, which can be found at: http://www.sgi.com/products/appsdirectory.dir/irix/products/v/955701.html.

2.6.6 Save TIN:

The TIN model is saved in the ".ver", ".tri" format. In particular two files, "***TIN.ver***" and "***TIN.tri***", are generated in the same directory where the executable is. Such a button is always enabled and it can be used to save the TIN model after the rotation, the normalization and contouring operations.

3 TUTORIAL

The GUI pipeline is shown, step by step, in the following.
Let us suppose that a model, for example a torus, is placed in the directory TESTS. We should have the two files: torus.ver and torus.tri.

The GUI starts with the command (written in a Linux terminal): ./reeb TESTS/torus and the following GUI will appear:

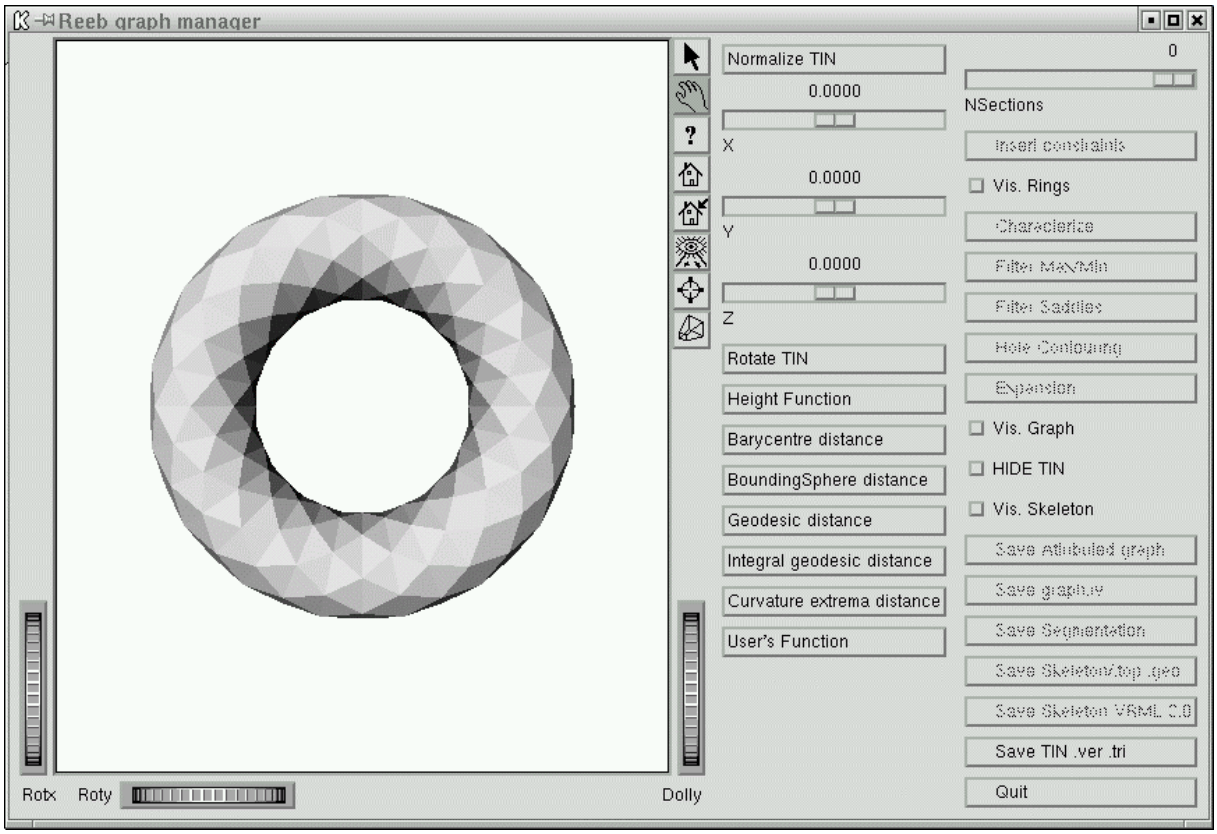
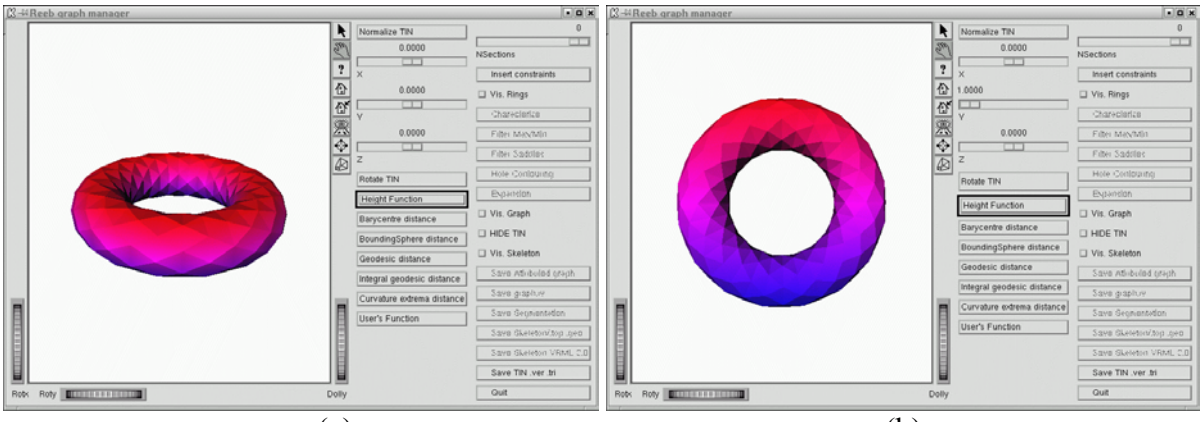
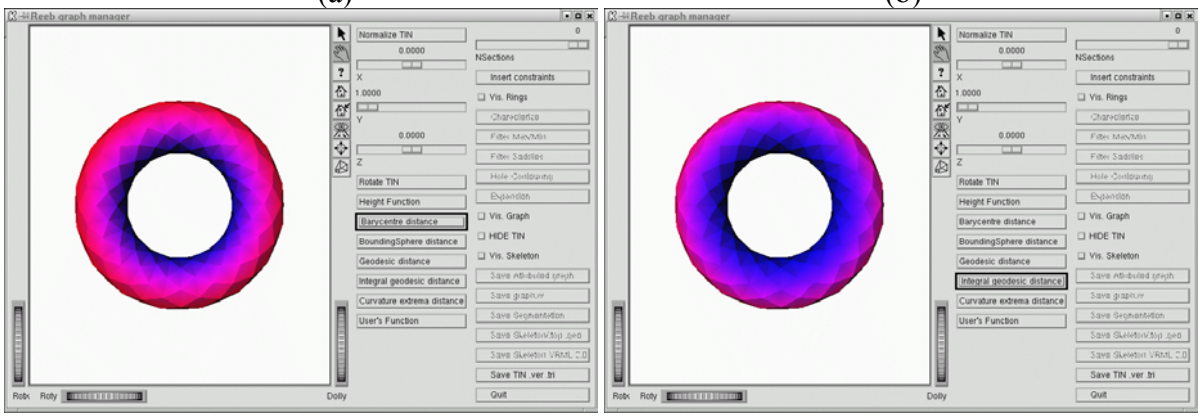


Figure 1: The initial appearance of SHREC.



(a) (b)



(c) (d)

Figure 2: The GUI after some choices of the measuring function; (a) measuring function with respect to the canonical height direction, (b) height function after a rotation of the TIN (vector (0, 1, 0)); (c) distance from the barycentre; (d) the integral geodesic distance.

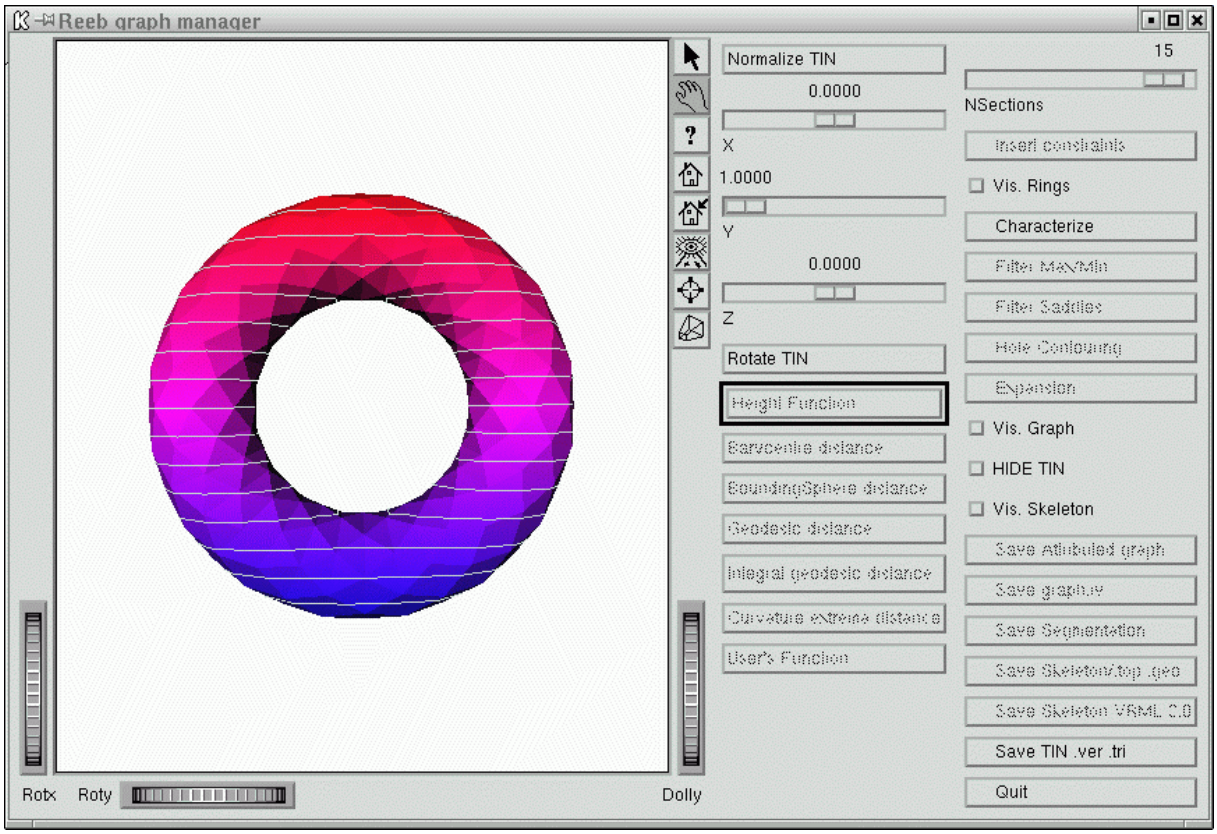


Figure 3: Insertion of 15 contour levels of the function in Figure 2(b) in the TIN.

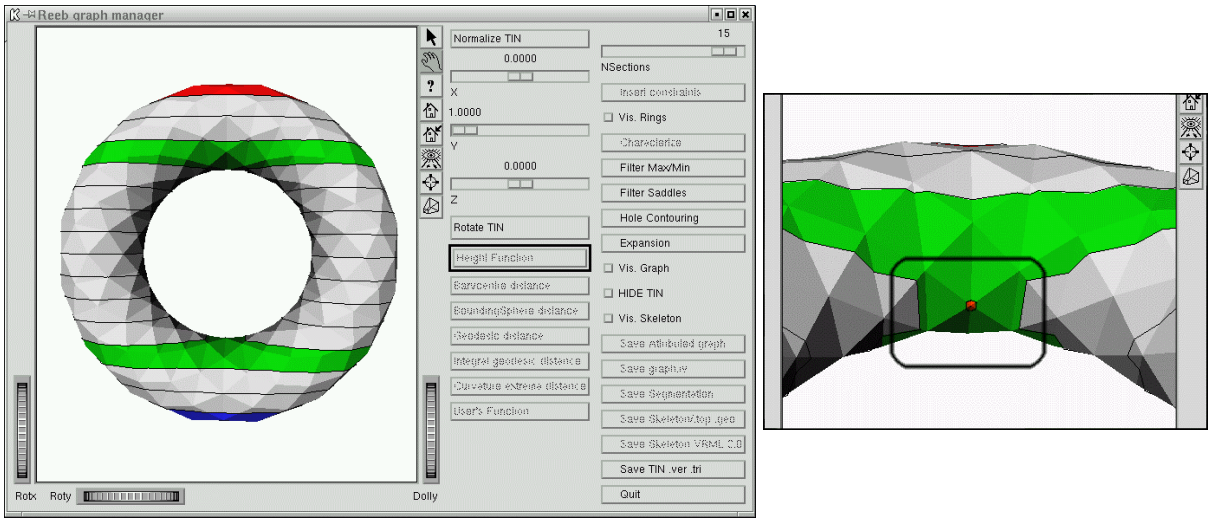


Figure 4: Surface characterization. In (b) is highlighted a minimum area that depends on the local shape of the triangles of the model but is not significant in the global surface characterization

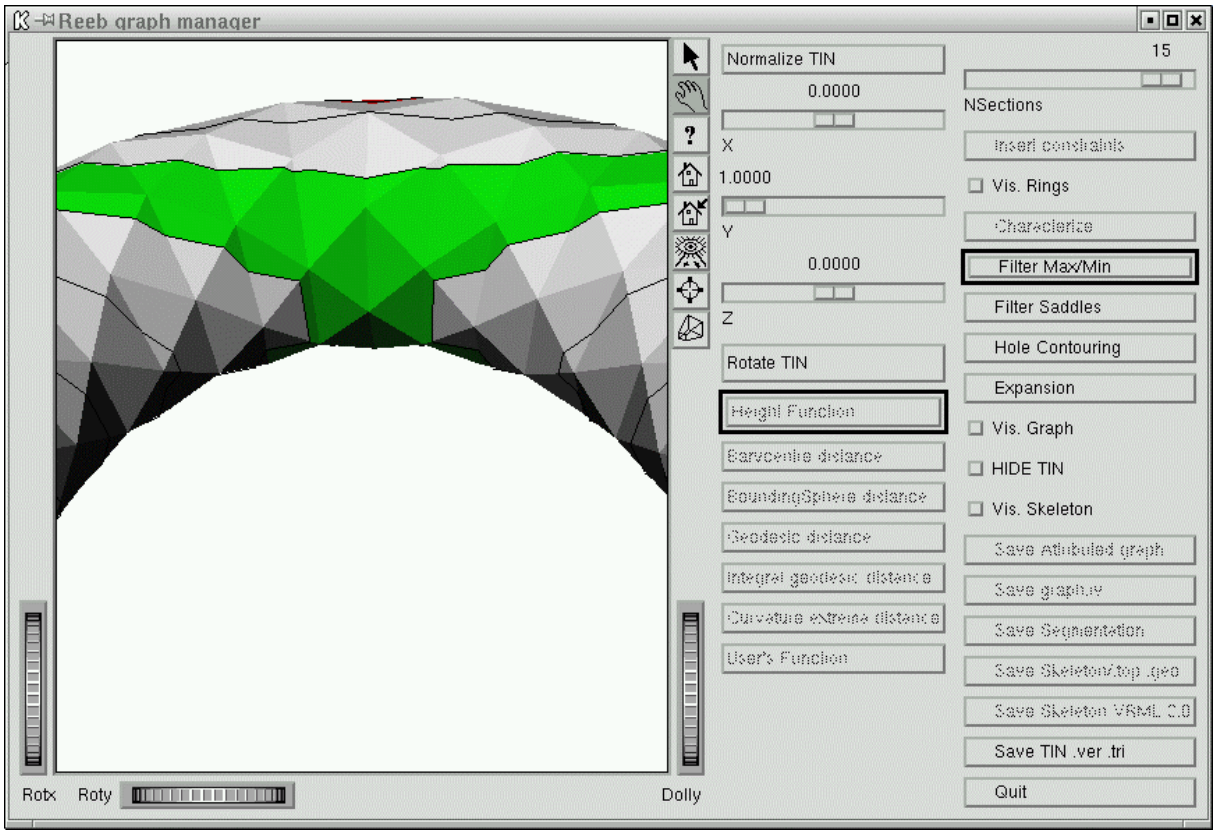


Figure 5: Filtering of non-significant maxima and minima.

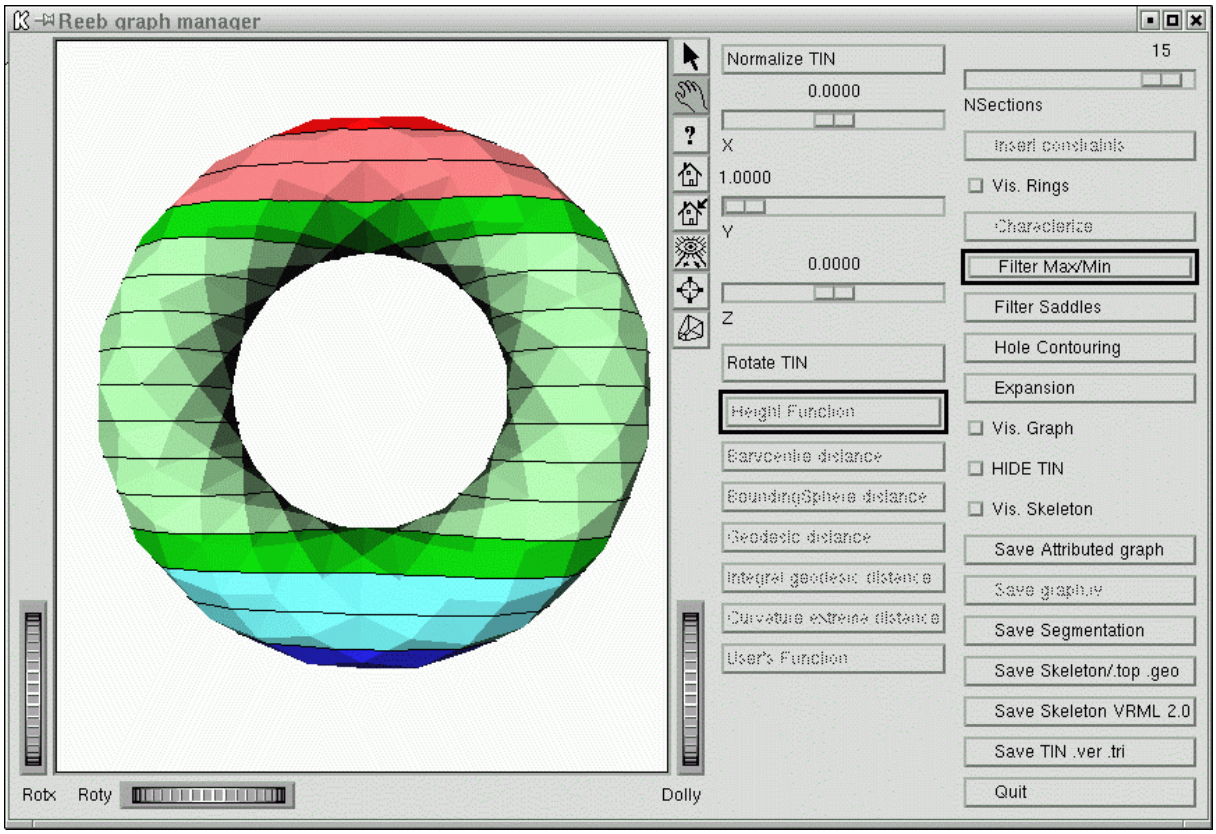


Figure 6: Expansion and visit of all triangles.

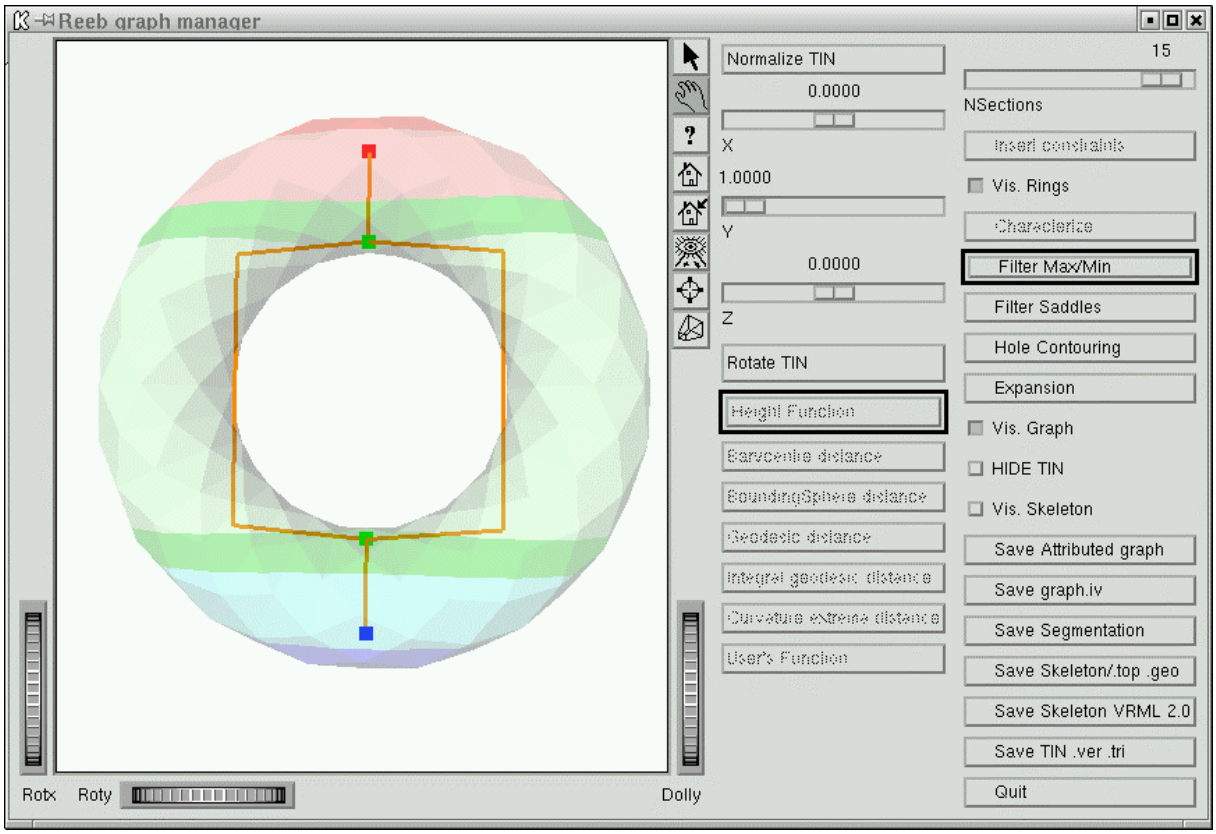


Figure 7: Visualization of the Reeb graph after the contour levels have been hidden.

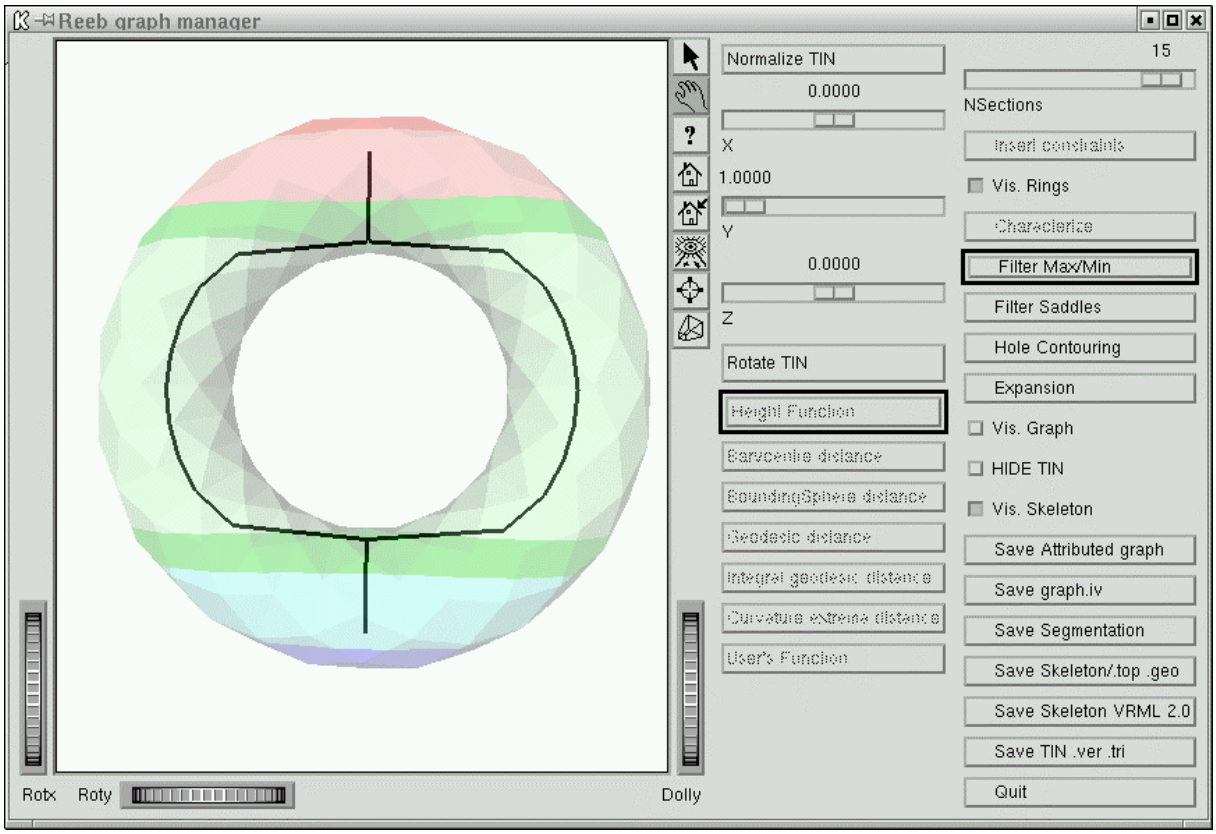


Figure 8: Visualization of the skeleton associated to the graph

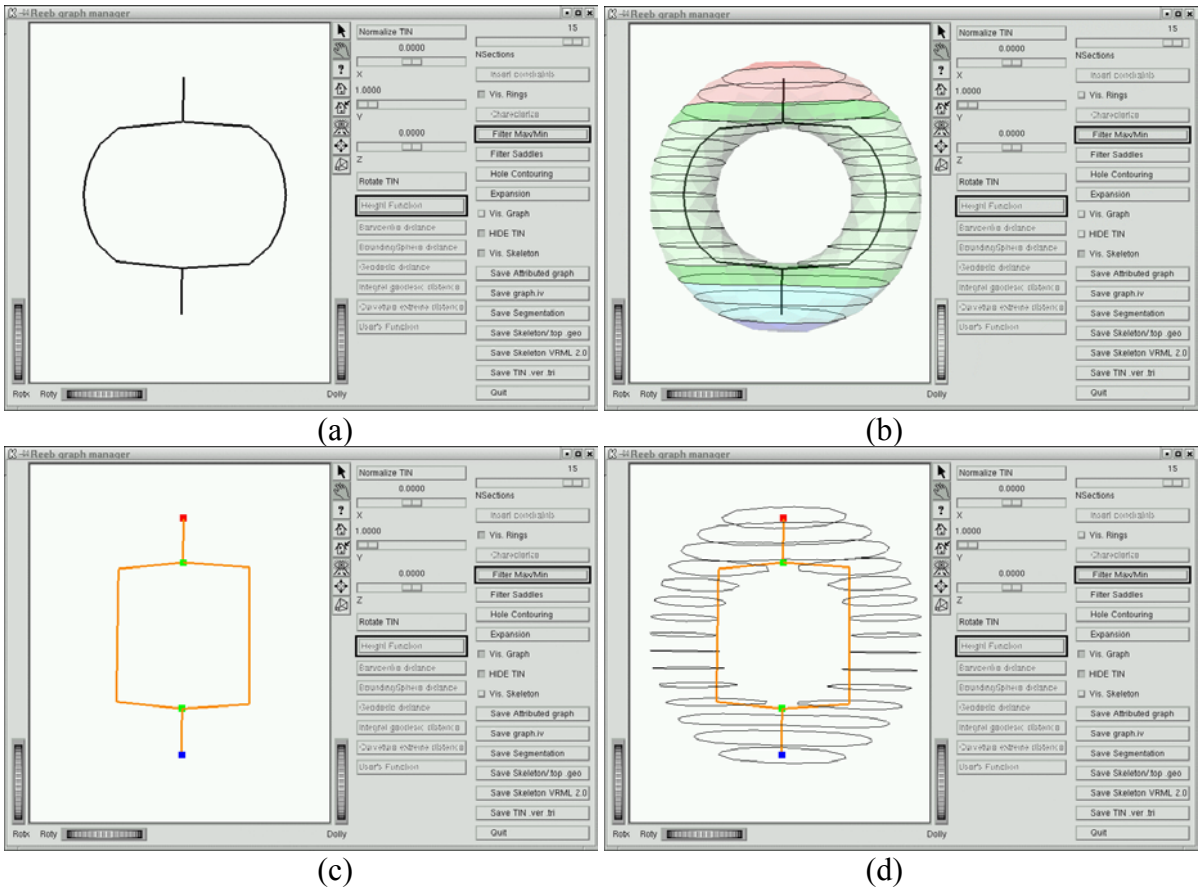


Figure 9: Other visualizations obtained hiding the TIN (a), showing the contour levels (b), etc.

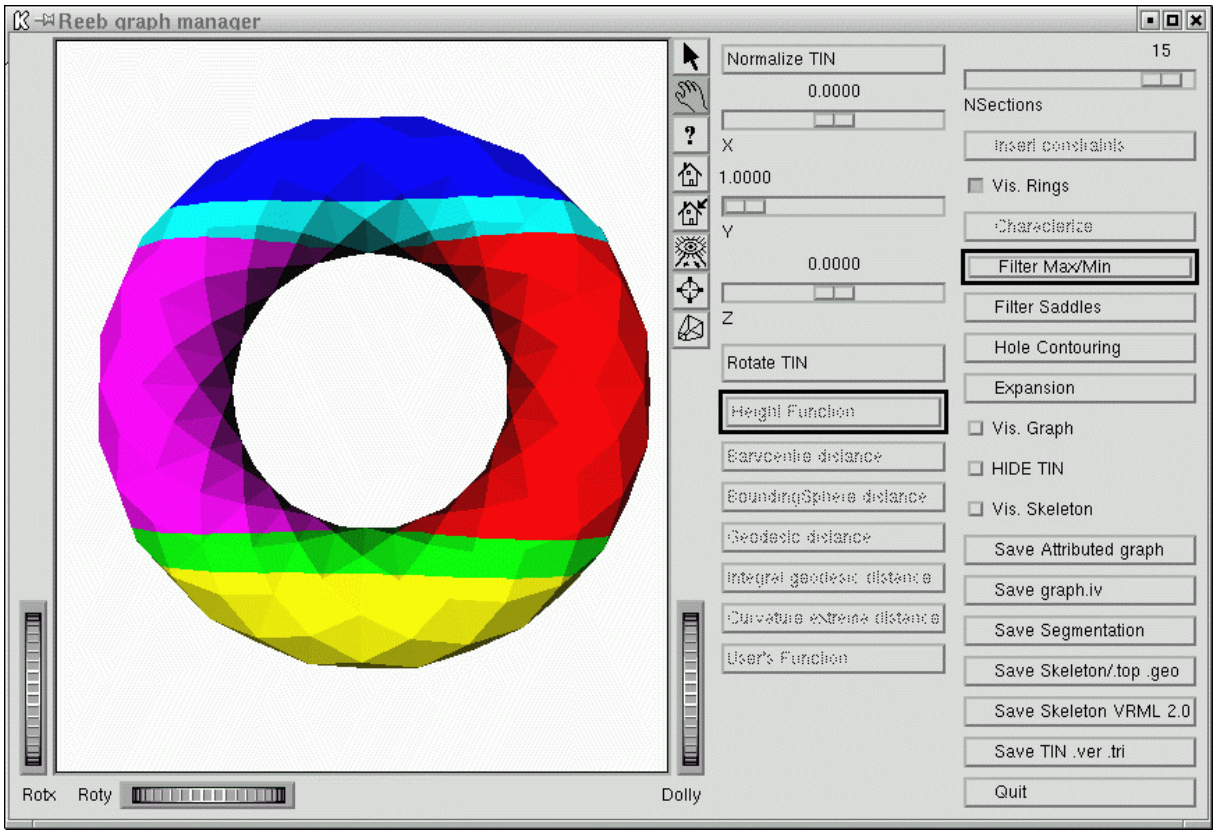


Figure 10: The appearance of the model after the surface segmentation has been saved in the file "SegmentedTIN.off".

4 REFERENCES:

[1] M.Attene, S.Biasotti and M. Spagnolo, “Shape Understanding by Contour-driven Re-tiling”, The Visual Computer, 19(2-3):127-138, 2003

[2] S. Biasotti, S. Marini, M. Mortara and G. Patanè, “An overview on properties and efficacy of topological skeletons in shape modelling”, In Shape Modelling and Applications 2003, IEEE Press, Seoul, pp. 245--254, 2003

[3] S. Biasotti, “Computational Topology Methods for Shape Modelling Applications”. Ph.D. thesis, Dept. of Mathematics, University of Genova, 2004

[4] M. Hilaga, Y. Shinagawa, T. Komura and T. Kunii, “Topology matching for fully automatic similarity estimation of 3D shapes”, Proc. SIGGRAPH 2001, pp. 203-212

[5] F. Lazarus and A. Verroust, “Level Set Diagrams of Polyhedral Objects”, In Solid Modeling and Applications, Ann-Arbour, 1999

[6] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno and J. Rossignac, “Blowing bubbles for multi-scale analysis and decomposition of triangle meshes”, Algorithmica, Special Issue on Shape Algorithmics, Springer-Verlag. 2004.

[7] M. Mortara and G. Patanè, Shape-Covering for Skeleton Extraction, Int. J. of Shape Modelling, 8(2):245--252, 2002