Project Elective December 14, 2022

Reeb Graph Computation And Visualization

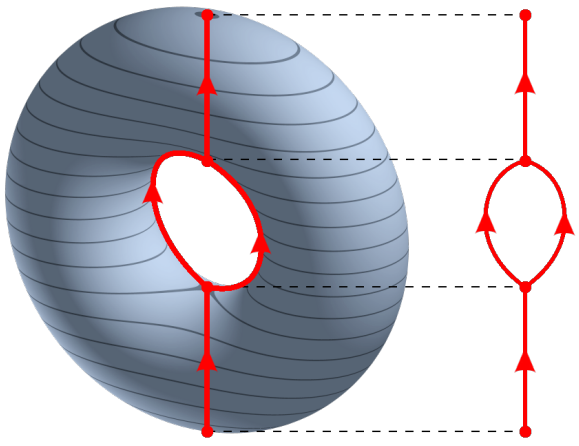
Anurag Singh Naruka (IMT2020093)
Ujjwal Agarwal (IMT2020128)
Instructor: Prof. Amit Chattopadhyay

Abstract: This is the summary of our understanding of the research paper "Output-Sensitive Construction of Reeb Graphs" by Harish Doraiswamy and Vijay Natarajan [1]. Here we will discuss the necessary definitions and working of the algorithm proposed in this paper with its key aspects. We will also discuss how we visualize the obtained Reeb Graph in a 3-D space as a contour tree Radial Tree graph layout and a tool made to interact with the reeb graph for analysing it.

1 Introduction

A Reeb graph is a mathematical object reflecting the evolution of the level sets of a real-valued function on a manifold. The Reeb graph of a scalar function is obtained by mapping each connected component of its level sets to a point. For 2 manifolds Cole-Mclaughlin et al proposed an algorithm that efficiently stores and manipulates the connected components of level sets that lead to fast construction of Reeb graph in O(nlogn) by maintaining the level sets using dynamically balanced search trees. There are other efficient Reeb graph computation techniques, however they make certain assumptions based on the graphs.

For higher manifolds or non-manifold, Doraiswamy and Natarajan stored the connected components of level sets using dynamic connectivity data structures resulting in an algorithm that computes the Reeb graph of a scalar function. It presents an efficient two-step algorithm for computing the Reeb graph of a piecewise-linear (PL) function in O(n + l +t logt) time, where n is the number of triangles in the input mesh, t is the number of critical points of the function, and l is the total size (number of edges) of all critical level sets.



2 BACKGROUND

2.1 Level Set

A level set of a real-valued function f of n real variables is a set where the function takes on a given constant value c, that is:

$$L_c(f) = \{(x_1, \ldots, x_n) \mid f(x_1, \ldots, x_n) = c\}$$

2.2 Contour Tree

A contour is a single connected component of the level set.The contour tree [2] is a loop-free Reebgraph that tracks the evolution of contours. Each leaf node is a minimum or maximum; each interior node is a saddle; each edge represents a set of adjacent contours with isovalues between the values of its two ends. There is a one-to-one mapping from a point in the contour tree (at a node or in an edge) to a contour of the scalar field.

2.3 Morse Function

A function f : M → R is a Morse function iff the following conditions are met:

1. None of f's critical points are degenerate.
2. No two of f's critical points share the same function value.

2.4 Radial Tree Layout

The Radial-Tree Layout arranges nodes in a circular layout, positioning the root node at the center of the graph and the child nodes in a circular fashion around the root. Sub-trees formed by the branching of child nodes are located radically around the child nodes. In this paper we are visualizing the side view of Radial Tree.

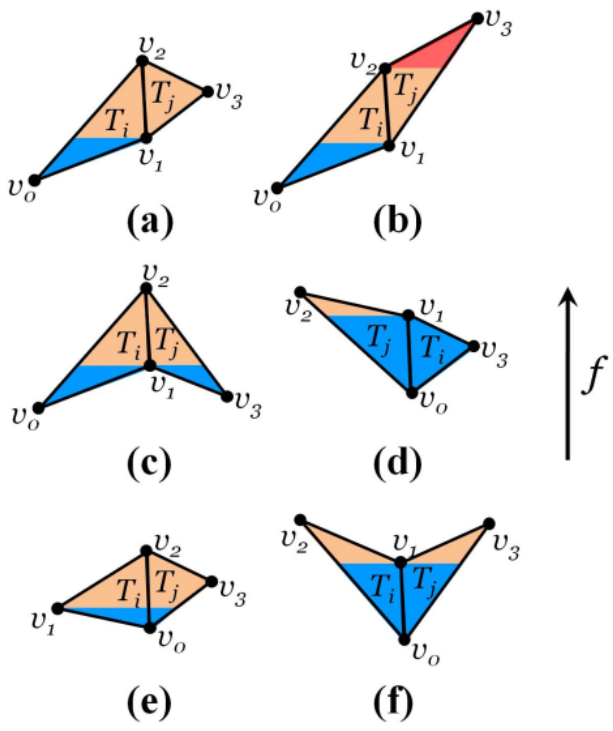
3 Reeb Graph Algorithm

This algorithm computes a Reeb graph of a piecewise linear function f defined on a 3-manifold. This consist of two steps: firstly we will locate all the critical points in the domain and sort them based on their function value and then identify pairs of critical points that define **cylinders** and insert the corresponding arc in Reeb graph.

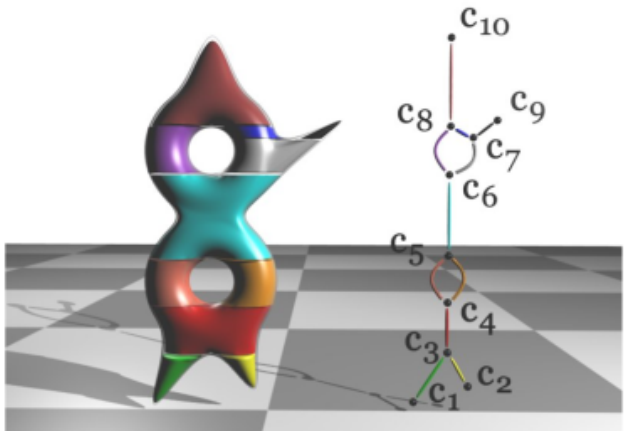
Cylinders: Collection of mesh triangles. A cylinder bounded by two critical level set components is represented by triangles that contain the intermediate level set components.

As discussed above Reeb graph is obtained by mapping each connected component of its level sets to a point. Here nodes of Reeb Graph will be mapped to components of critical level sets and arcs will be mapped to equivalence classes of regular level set components.

A dual graph is introduced to store triangle adjacencies and helps implicitly track level set components of individual cylinders. This graph is a directed graph GLS (V,E), called the LS-graph. It has nodes V = {t_1,t_2,...,t_n} corresponds to the n triangles {T_1,T_2,...,T_n} in the input mesh.



In the given triangulation of the input mesh, there can only be six possible configurations of adjacent triangles. The LS-graph contains an edge from t_i to t_j in all cases except the forbidden configuration in (f). Edges in the graph are directed toward the node with a higher cost



Connecting the critical points: Here of the two-step algorithm computing the Reeb graph of the height function defined on a solid 2-torus. This model has ten critical points, including two minima, two maxima, and six saddle points. The critical points are first sorted in increasing order of function value. Let $c_1, c_2, ..., c_{10}$ be the ten critical points in sorted order. Beginning with a triangle in the upper star of c_1, the algorithm traces the green cylinder to reach L_3 and inserts (c_1, c_3) into the Reeb graph. Then the search from c_2 also reaches L_3, but a different component as compared to the previous trace. So (c_2, c_3) is inserted into the Reeb graph. The upper star of c_4 has two components. A search is initiated from each component to obtain the two parallel arcs (c_4, c_5) of the Reeb graph. While tracing the cylinder from c_6, the search procedure reaches a triangle with cost greater than f7 that does not belong to L_7. The search procedure next reaches $L8$ and the arc (c_6, c_8) is inserted into the Reeb graph.The Reeb graph of the input function is computed after all critical points are processed.

4 Implementation

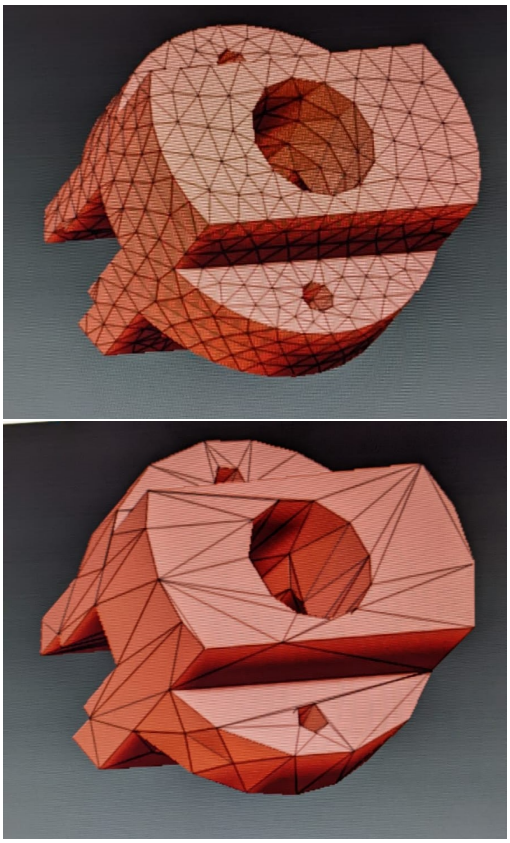
4.1 Mesh Simplification

We then explored some mesh simplification tools in order to remove unnecessary processing. Mainly we found two tools -

1. **Blender**
2. **ZBrush**

In blender we were able to perform mesh simplification after following some tutorials, but the process sometimes failed due to unknown reasons.

Zbrush on the other hand, was much easier to use. ZBrush uses **quadratic decimation** to perform mesh simplification which does not affect or change the topology. Hence was ideal for our use.



4.2 Conversion of files

We have been working with different types of files. For making the reeb graph we need to input the file in OFF format. We have written a python program to convert gts to off format and for other formats you can use this link.

Now talking about off file format.

- Optional first line containing "OFF".
- Next line specifies the no. of vertices (nv) followed by the number of triangles (nt) (space seperated)
- The next nv lines contains x y z f where x, y z specify the co-ordinates of the vertex and f specifies the function value.
- The next nt lines has 3 v1 v2 v3 where v1, v2 and v3 are the vertex indices of the vertices that form the triangles.

In the third point as mentioned above we have to give functional value that will correspond to the function based on which the algorithm will be computing critical points and compute a reeb graph. Consider that function as a height function on z-axis, so in our off file format the nv lines will look like x y z z(this z for function value) then algorithm will compute a reeb graph based on the given height function. We have written a file named append_height_z to give a functional value to any off file(available on github). **These functional value can also take scaler values.**

4.3 Observations from Reeb Graph file

Now we have the off file with some functional value we will give this input to our algorithm and get the output file(.rg file) which basically is our reeb graph.

[.rg file for Torus]

```
4 4
0 -54.36 MINIMA
13 -32.1389 SADDLE
150 3.5702 SADDLE
145 25.7913 MAXIMA
0 13
13 150
150 145
13 150
```

In first line of the .rg file there are two things 1st number is the number of critcal points say M and 2nd number denotes the number of edges(or arcs) say N in the Reeb graph. Next following M lines give three things, indices of the critical points followed by function value and then the type of critical point. And there are N more lines which tell us about the edges (or arcs) between the two vertices.

From the above example of the .rg file of Torus, We can see that the vertex 13 and 150 are SADDLES and there is a connection between these two SADDLES which is repeated (3rd last line and last line of Example) this tells us that there are two arcs between vertex 13 and 150 which should be there as it is a Torus. So for making the loops in Radial Tree Layout if a connection is repeated we will add a loop between these points seprately.

All critical points will be either minima or maxima will exactly be a part of one edge in the graph which means that if me make a graph from these N connections(edges or arcs) the degree of all minima or maxima will be one i.e all these points will be the leaf nodes of the graph and all other nodes (except the root node) will be a SADDLE.

4.4 Radial Tree Layout Of Reeb Graph

Now as we have the .rg file we can use any layout scheme and visualize it. Here we will use the Radial Tree layout to visualize the Reeb graph.Radial Tree layout scheme is not easy because of the loops present

in Reeb Graph.We overcome this difficulty by designing a four step layout scheme:

1. Extract a spanning contour tree of the Reeb graph.

2. Compute a branch decomposition of this spanning tree.

3. Use a Radial Tree layout scheme to embed the spanning tree in 3D

4. Add the non-tree arcs to the layout.

We have written a layout.sh file to run the LayoutReebGraph.java files that calls the four functions as mentioned above.We will input our Reeb Graph file to the layout.sh and we will get a text file which is Radial Tree Layout for the given off file.

Output File of Radial Tree Layout for Torus

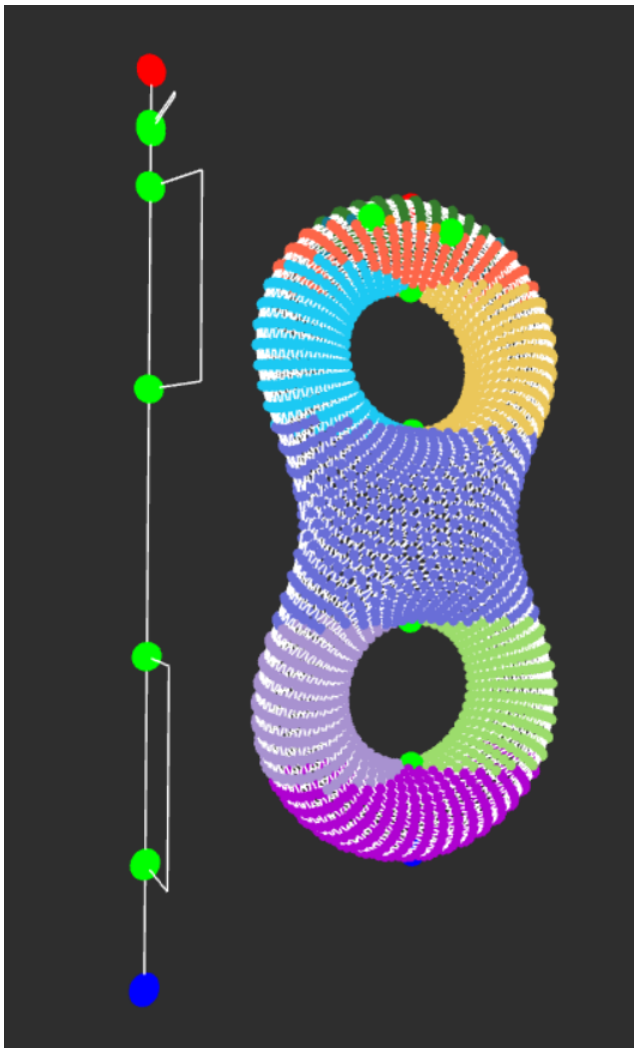
```
0 0.0 -2.0 0.0
13 0.0 -0.89104235 0.0
150 0.0 0.89104223 0.0
145 0.0 2.0 0.0
```

The layout file will have all the critical points and its location (x y z co-ordinate of that point).

4.5 Visualization Of Radial Tree Layout

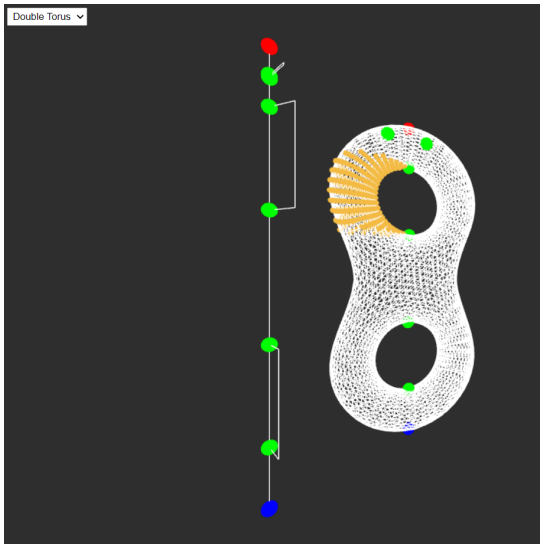
The Radial Tree Layout has been viasualized using the Three.js Library which is an application programming interface used to create and display animated 3D computer graphics in a web browser using WebGL.

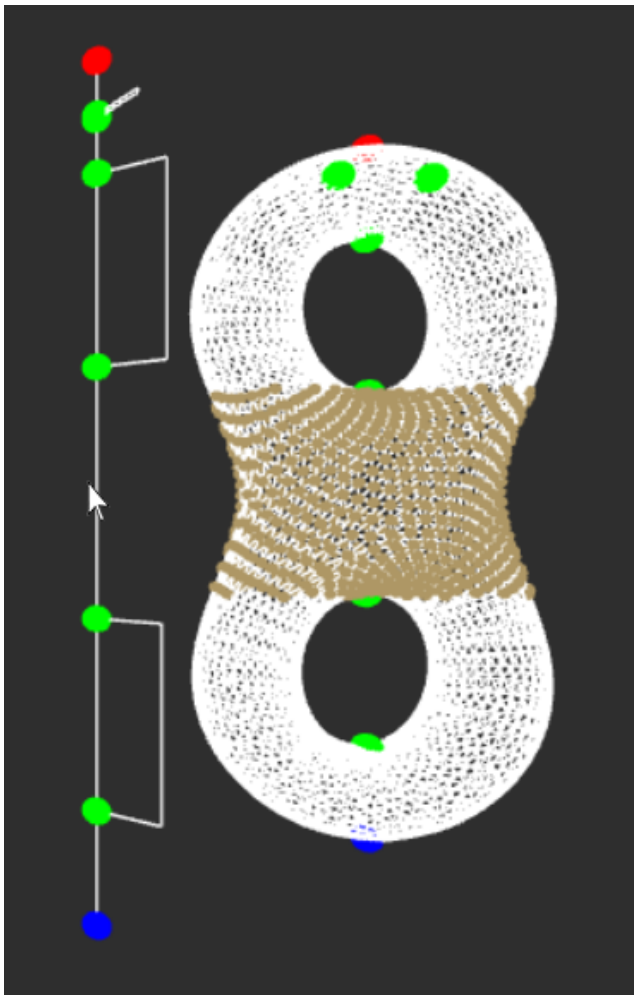
Here we create a tool to interact with the reeb graph of a manifold to better understand and analyze it. The reeb graph helps us to visualize the critical points, red for maxima, green for saddle and blue for minima. The edges represents all the regular points of the shape.



4.6 Examples

In this tool when an edge of the reeb graph is hovered over with the pointer then the corresponding points of the level set are highlighted on the mesh of the manifold.





References

[1] Harish Doraiswamy and Vijay Natarajan. "Computing Reeb Graphs as a Union of Contour Trees". In: *IEEE transactions on visualization and computer graphics* 19 (Apr. 2012). DOI: 10.1109/TVCG.2012.115.

[2] Keqin wu and Song Zhang. "A contour tree based visualization for exploring data with uncertainty". In: *International Journal for Uncertainty Quantification* 3 (Jan. 2013), pp. 203–223. DOI: 10.1615/Int.J.UncertaintyQuantification.2012003956.