# Isosurface Computation and Rendering Tool

**Vishal Peter, Srishty Suman**
M. Tech, CSA

## 1 Problem Formulation

For 2D or 3D meshes that represent the domain of a continuous function of real values, the contours or iso-surfaces of a specified scalar value are an important way to visualize the function. To find such contours, a seed set can be used as the starting point from which the traversal of the contours can begin. In this project, our aim is to extract iso-surfaces of 3D meshes containing scalar field visualization data and render them for different scalar values using contour propagation using seed sets and display them in an interactive, user friendly manner.

## 2 Related Work

Previous works which we have studied are listed here. [4] [3][1]. We have implemented marching cube algorithm using [2] as reference.

## 3 Work Completed

### 3.1 Real Time Contour Propagation

We have implemented a pre-processing based algorithm for calculating contours in real time for different scalar values. Display is supported for a predefined list of scalar values(say $S$). By default, the display tool supports all scalar values between 0 to 1 with a step-size of 0.01(ie {0.01, 0.02 .. 1.00}). Additionally, the user can also specify more specific scalar values in a config file for a finer control over isosurfaces to be displayed. We have made our code publicly available at https://github.com/viz27/visualize3D. The algorithm is described below.

- Let $L_i$ denote the list of grid cells that contain scalar value $i$.
- First, do a sweep of all the cells in data grid and add the index of each grid cell to the list $L_i$ if scalar value $i$ passes through the grid cell. For eg. if scalar values $0.01$ and $0.02$ passes through grid cell of index $1000$, then $1000$ will be added to $L_{0.01}$ and $L_{0.02}$.
- Once we have the list $L_i$, we can directly look at the grid cells in the list and extract contours from those cells as required. We call this mode of operation **FAST Mode**. In this mode, index of all the intersecting cells for all the scalar values in $S$ need to be kept in memory. Hence, it is memory inefficient, but contour propagation is fast.
- We also describe another mode called **SEED Mode**, where seed sets are calculated for every scalar values in $S$.
- Given cell IDs $a, b \in L_i$, we can combine $a$ and $b$ to a single set if $a$ and $b$ are neighbors in the data grid. Checking all entries in $L_i$ in this manner and combining neighbors to single set, items in $L_i$ will be divided into disjoint sets and elements in each set will belong to the same connected component in the isosurface of scalar value $i$. This combination can be done efficiently using union-find algorithm. Taking a single representative item from each set in $L_i$, we will get a small number of seed cells from which all the other cells can be identified via propagation.

Submitted to Prof. Vijay Natarajan.

- Instead of calculating seed sets for each scalar value one after another, we describe a method to calculate it in one sweep of the entire grid cells. For a grid cell with ID j, consider all its neighbors(up to 28 neighbors for a cubic cell) and whenever cell j and neighbor k belong to the list $L_i$ (ie. scalar value i passes through both cells j and k), merge them in $L_i$ (using union find algorithm). After sweeping over the entire grid cells in this manner, each $L_i$ will be split into disjoint sets from which seed cells can be extracted.
- Once we have the seed set for each scalar value $i \in S$, this can be saved and for subsequent runs, this can be retrieved and contours can be calculated based on the seed set, eliminating the pre-processing phase in subsequent runs.

### 3.2 Interactivity

1. "Space key input" to stop and start rotating the iso-surface.
2. Trackball to rotate the object along $x$ and $y$ axes.
3. Scrollbar for changing the camera position and re-rendering simultaneously.
4. Scrollbar for changing the iso-values and re-rendering.
5. Keyboard input to specify new scalar values.

## 4 Experiments and Results

Here we report the processing time and number of vertices generated per iso-surface for 4 different scientific visualization datasets. The numbers are shown for the program when run in FAST Mode.

| | Preprocessing Time | | | Contour Extraction Time | | | | Number of vertices in isosurface | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | data read | generate cells | total | 0.01 | 0.1 | 0.2 | 0.5 | 0.01 | 0.1 | 0.2 | 0.5 |
| teapot_256x256x178_uint8 | 1s | 1s | 2s | 0.32s | 0.24s | 0.23s | 0.007s | 2.9M | 2.2M | 2.1M | 72K |
| bonsai_256x256x256_uint8 | 2s | 1s | 3s | 1.1s | 0.58s | 0.16s | 0.17s | 11.5M | 5.8M | 1.7M | 1.9M |
| csafeheptane_302x302x302_uint8 | 3s | 2s | 5s | 0.35s | 0.39s | 0.24s | 0.15s | 3.0M | 3.4M | 2.2M | 1.5M |
| stent_512x512x174_uint16 | 8s | 2s | 10s | 0.69s | 0.38s | 0.31s | 0.02s | 7.2M | 4.0M | 3.3M | 0.3M |

Table 1: Performance of visualization tool in FAST Mode

## 5 Discussion

As seen from Table 1, our tool is quite fast for both pre-processing as well as real time contour extraction for Medium sized datasets when run in FAST Mode. When run in SEED Mode, the pre-processing and contour extraction times are greater. This is likely due to union-find algorithm overhead for seed cell extraction and Breadth First Search overhead for contour propagation.

## 6 Contributions

- Vishal Peter: Algorithm and some interactivity parts.
- Srishty Suman: Interactivity part.

## References


[1] A. Bock, H. Doraiswamy, A. Summers, and C. Silva. Topoangler: Interactive topology-based extraction of fishes. *IEEE transactions on visualization and computer graphics*, 24(1):812–821, 2017.

[2] P. Bourke. Polygonising a scalar field, http://www.paulbourke.net/geometry/polygonise/, 1994.

[3] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75–94, 2003.

[4] M. Van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proceedings of the thirteenth annual symposium on Computational geometry*, pages 212–220, 1997.