TRANSFER FUNCTION GENERATION FOR DIRECT VOLUME RENDERING

B.Tech Project Report
submitted by

M. Aishwarya

Roll No: COE14B020

Under the Guidance of

Dr. N. Sadagopan

in partial fulfilment for the award of the degree

Bachelor of Technology

in

Computer Engineering

Indian Institute of Information Technology Design and Manufacturing

Kancheepuram, Melakottaiyur, Chennai-600127

April 2018

Bonafide Certificate

This is to certify that the btech project titled "Transfer Function Generation For Direct Volume Rendering" submitted by M. Aishwarya(COE14B020) to the Indian Institute of Information Technology Design and Manufacturing Kancheepuram, is a bonafide record of the project work done by her under my supervision. The contents of the report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. N. Sadagopan

Project Guide
Assistant Professor
Indian Institute of Information Technology Design and Manufacturing
Kancheepuram
Chennai 600127
India.

Place: Chennai
Date: 25/04/2018

Abstract

Visualization is an important tool in the field of medicine and non–destructive testing(NDT). Direct volume rendering is a powerful technique for visualizing large three-dimensional data sets. A direct volume renderer requires every sample value to be mapped to opacity and a color. This is done with the help of transfer function. The opacity transfer function is usually designed as piecewise functions for flexibly displaying each section of the whole object. Hence, it is critical to determine the piecewise nodes in the transfer function. Setting the transfer functions is difficult, unintuitive, and slow process.

This project is about implementing a transfer function generator which works at interactive speed. The algorithm for transfer function generator is designed as a two-step process. The first step focuses on the computation of boundary characteristics of the input volume. In this step, the algorithm finds the data values corresponding to the object. In order to visualize different parts of the object, data values corresponding to the object are classified into multiple subsections using accumulative probability distribution of volume data. Then, a threshold data value is computed in each subsection. These threshold data values act as piecewise nodes of the opacity transfer function which is given to Direct Volume Rendering algorithm. The user can set opacity to each threshold data value using sliders and visualize the changes in the rendered volume instantly.
The algorithm has been implemented in C++ with the support of ITK[Insight Toolkit] and VTK[Visualization Toolkit] in visual studio. The C++ program takes 3D volume as input and generates a transfer function which is used by the direct volume rendering algorithm to produce the final rendered volume as output.

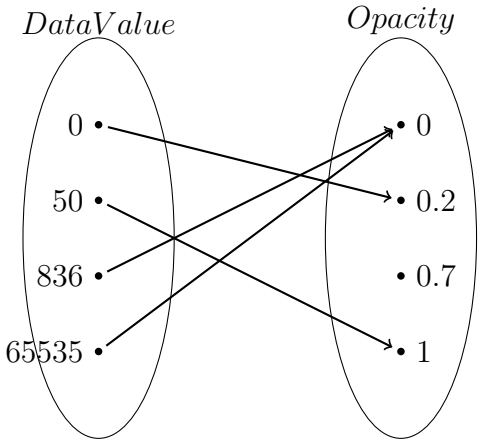
This project is done in collaboration between IIITD&M Kancheepuram and Lucid Pvt.Ltd. The industrial CT scan volumes provided by the company are given as inputs to the final software developed through this project. And the final software tool is intended to be contributed to the VTK open source community.

Contents

1 Introduction

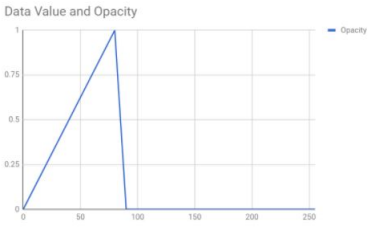
Non-destructive Testing(NDT) plays an important role in assuring the quality of a system without destroying its serviceability. Industrial Computed Tomography is one of the most important NDT techniques which produces three-dimensional(3D) representations of the scanned object both externally and internally. The 3D volumetric data produced by Computed Tomography(CT) scanner helps experts to detect flaws inside a part such as porosity, an inclusion, or a crack. But experts analyse the 3D volumetric data on a two-dimensional(2D) computer screen, so the 3D dataset must be projected onto a 2D image plane to gain an understanding of the structure contained within the data. This process of projecting is done with the help of volume visualization techniques. One of the most important techniques for visualizing large amount of 3D data is Direct Volume Rendering(DVR). The 3D volume produced by the CT may also contain noise and other components which might not be of interest for the final user. So the DVR algorithm gives the user a chance to set the opacity of the area of interest to one and the opacity of the remaining region can be set to zero. This task of assigning optical properties such as color and opacity to original values of the data set being visualized is done with the help of transfer function. The opacity transfer function is usually designed as piecewise functions for flexibly displaying each section of the whole object. Hence, it is critical to determine the piecewise nodes in the transfer function.Unfortunately, finding good transfer functions is challenging and time-consuming. In this project, a transfer function generator for direct volume rendering is implemented.

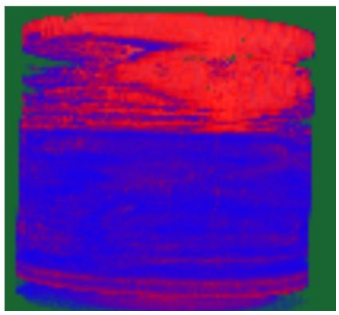
Mathematically, TransferFunction:(DataValue:[0,65535])→(Opacity:[0,1]).



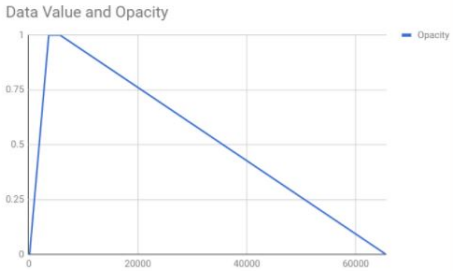
Savings box is rendered using different opacity transfer functions to emphasise the importance of having a good transfer function.

Data Value	Opacity
0.0	0.0
80.0	1.0
90.0	0.0
255.0	0.0

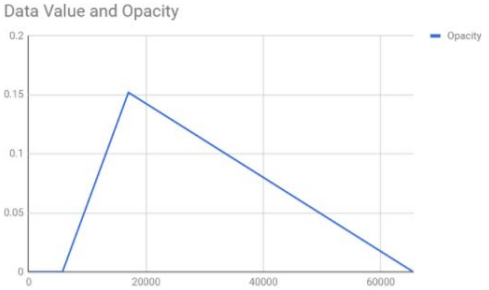




Data Value	Opacity
0.0	0.0
150.0	0.0
3662.0	1.0
5750.0	1.0
65535.0	0.0



Data Value	Opacity
0.0	0.0
150.0	0.0
5750.0	0.0
16998.0	0.152
65535.0	0.0



1.1 Background

Volume: A 3-dimensional data set is referred to as a volumetric data set or volume.
Voxel: It is the 3-dimensional (3D) equivalent of a pixel and the tiniest distinguishable element of a 3D object. However, like pixels, voxels do not contain information about their position in 3D space. Rather, coordinates are inferred based on their designated positions relative to other surrounding voxels.
Data Value: Each voxel has a scalar value. This scalar value is considered as data value. Here, in this case the scalar value is a 16 bit grey scale. Here, Data value range is [0,65535].
Volume Rendering: In scientific visualization and computer graphics, volume rendering is a set of techniques used to display a 2D projection of a 3D discretely sampled data set, typically a 3D scalar field.
There are two types of volume rendering methods:
Indirect Volume Rendering: This approach consists of two steps. The first one is an extraction of isosurface out of a dataset in accordance with certain threshold. Then the polygonal surface model can be visualized by any 3D engine or other visualization tools.
Direct Volume Rendering: It does not require any preprocessing. The data is visualized from an original dataset. It gives the algorithms an opportunity to modify the transfer function and threshold dynamically. Also, some of the approaches allow to visualize the internal structure of the dataset in semi-transparent way.
Direct volume rendering is practically the most powerful way to visualize volume data. The visualization has all the advantages of polygonal mesh models, and can be easily combined with them on the same scene. In addition, it is possible to cut a part of the model for an investigation of structures hidden by the object surface.For all these reasons, direct volume rendering is used in this project.
Transfer Function: Transfer functions make volume data visible by mapping data values to optical properties. The data values are mapped to color and opacity. Transfer function can be a simple ramp, a piecewise linear function or an arbitrary table.
Opacity is the measure of impenetrability to electromagnetic or other kinds of radiation, especially visible light. An opaque object is neither transparent (allowing all light to pass through) nor translucent (allowing some light to pass through). Opacity(α) range is [0,1].
Transfer functions use opacity to identify the values of the variable that are most interesting and other values can be made transparent. Color is used to highlight particular values, e.g. to distinguish them from other values.
Edge: Place of local transition from one object to another.

Directional Derivatives: In mathematics, the directional derivative of a multivariate differentiable function along a given vector v at a given point x intuitively represents the instantaneous rate of change of the function, moving through x with a velocity specified by v.

First Derivative or Gradient: An image gradient is a directional change in the intensity or color in an image. The gradient magnitude gives the amount of the difference between pixels in the neighbourhood (the strength of the edge).

Second Derivative: Differentiating the first derivative (gradient magnitude) gives us the second derivative.

Edge Detection: Finding the ideal edge is equivalent to finding the point where the derivative is a maximum (for a rising edge with positive slope) or a minimum (for a falling edge with negative slope).

How do we find maxima or minima of one-dimensional functions? We differentiate them and find places where the derivative is zero. Finding optimal edges (maxima of gradient magnitude) is thus equivalent to finding places where the second derivative is zero.

When we apply differential operators to images, the zeroes rarely fall exactly on a pixel. Typically, they fall between pixels. We can isolate these zeroes by finding zero crossings: places where one pixel is positive and a neighbour is negative (or vice versa).

Probability Distribution: A probability distribution is a statistical function that describes all the possible values and likelihoods that a random variable can take within a given range. This range will be between the minimum and maximum statistically possible values.

Cross Entropy: The cross entropy was proposed by Kullback. Let F = $\{f_1, f_2, ..., f_N\}$ and G = $\{g_1, g_2, ..., g_N\}$ be two probability distributions on the same set. The cross entropy between F and G is an information theoretic distance between the two distributions and it is defined by

$$\mathrm{D(F,G)} = \sum_{i=1}^{N} f_i log(\frac{f_i}{gi})$$

Slider: A slider or track bar is a graphical control element with which a user may set a value by moving an indicator, usually in a horizontal fashion. Here, the user can set the opacity of a particular data value with the help of sliders.

2 Literature Review

There has been a great amount of research devoted to transfer function design that is an indispensable part of volume visualization. Transfer functions are particularly important to the quality of direct volume-rendered images. Setting the transfer functions is difficult, unintuitive, and slow process. The transfer functions for 3D visualization can be set manually or there are several image and data driven approaches for automatic and semi-automatic generation of transfer function.

2.1 Manual Generation of Transfer Function

Interactive volume rendering is a powerful tool for visual exploration of volumetric data. Transfer functions for color and opacity are manually generated using some kind of visual editor. A transfer function can be a simple ramp, a piecewise linear function or an arbitrary table. Manual transfer functions are generated in an iterative process and are based on heuristics, specialized knowledge and personal taste, the whole assignment procedure is rather time-consuming and hardly reproducible.

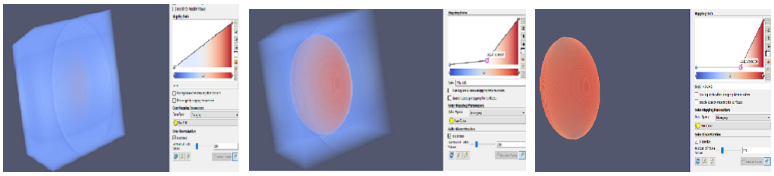


Figure 1: Setting the transfer function manually in Paraview.

2.2 Semi-automatic Generation of Transfer Function

The Semi-automatic generation methods can be divided into image-driven and data-driven methods.

Marks et al.[1] have proposed the concept of design galleries for setting image galleries in general.This example of a semi-automatic image-driven method generates a huge amount of images with different parameter settings, allowing the user to select the image with the optimal visual result. He et al.[2] employed a supervised genetic algorithm approach to find the best TF with either the user or the system itself, automatically ranking generations to identify the fittest member for the next iteration. The applicability of these image-based

approaches is limited, since they are neither fast nor purposive, and thus not efficient enough for clinical routine.

Data-centric TFs define visual properties based on the information of voxels. Fang et al. [3] have described a data-driven semi-automatic algorithm. In their implementation, they first apply a transformation to the entire dataset using image enhancement and boundary detection operations, followed by a simple 1D linear ramp to color the dataset. While focusing on material boundaries in datasets, Kindlmann et al.[4] proposed an interesting data-driven method which considered the 3D histogram volume defined by attribute value, first and second directional derivative. In this space, they have defined a model that has classified boundaries and then have used this model for automatic opacity function generation. Jorg et al.[5] presented an efficient technique for the construction of LH Histograms which generates transfer functions in real time. This paper takes Kindlmann's paper[4] as base idea. Tzeng et al.[6] propose using the iterative self-organizing data analysis technique to find the clusters in 2D histogram space. With this algorithm, the user can choose opacity and color for each cluster. Ip et al.[7] demonstrates an algorithm for multilevel segmentation of the intensity gradient 2D histogram that allows the user to select the most appropriate segments hierarchically for which they can generate separate TFs.

2.3 Automatic Generation of Transfer Function

The idea of using Morse theory to automatically decompose the feature space of the volume into a set of cells has been presented by Wang et al.[8] Using cell separability, cells that potentially contain important features are retained, whereas those most likely originating from noise are rejected.

Methods that eliminate the human from the exploration process are not suggested because visualization is not just about generating pretty pictures. It's also a vehicle of exploration by which the observer comes to understand the data. We can think of automatic techniques that generate images that fulfil the observer's expectations,but aren't necessarily true to the nature of the data. By observing all the above methods, we have chosen semi-automatic generation of transfer functions for direct volume rendering.

Mode of generation of transfer function: Semi-Automatic Generation

Reason: Automatic methods that eliminate people from the exploration process are not suggested and manual methods are very slow. So semi-automatic generation method is chosen. And image-driven approaches are neither fast nor effective so data driven approach is chosen.

3 Problem Statement

-Finding appropriate transfer function for direct volume rendering is a difficult problem because of the large amount of user experimentation typically involved.
-The main goal of this project:

- Formulating a method to generate transfer functions for direct volume rendering and implementing it on CPU.
- Developing a Transfer Function Generator that works at interactive speeds.

4 Boundary Model

We employ the boundary model presented by Kindlmann and Durkin[4]. They presented a mathematical model which assumes that at the boundary, objects have a sharp, discontinuous change in the physical property measured by the values in the dataset. The following equation describes the ideal boundary data value as a function of position:

$$\text{v} = \text{f(x)} = \text{v}_{min} + (\text{v}_{max} - \text{v}_{min})\frac{1+erf(\frac{x}{\sigma\sqrt{(2)}})}{2}$$
$$\text{where erf(x)} = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$$

Since erf() is centered around origin, the boundary function f(x) is also centered around where position x is zero. erf()'s range has been shifted and scaled so that range of f(x) is between v_{min} and v_{max}. As x approaches negative infinity, f(x) approaches v_{min} and as x approaches positive infinity, f(x) approaches v_{max}. At x=0, the middle of the boundary, f(x) is half-way between v_{min} and v_{max}. The first and second directional derivates of f along the gradient direction are as follows:

$$f'(x) = \frac{v_{max}-v_{min}}{\sigma\sqrt{2\pi}}e^{-\frac{x^2}{2\sigma^2}}$$
$$f''(x) = -\frac{x(v_{max}-v_{min})}{\sigma^3\sqrt{2\pi}}e^{-\frac{x^2}{2\sigma^2}}$$
$$\implies \frac{f''(x)}{f'(x)} = -\frac{x}{\sigma^2}$$
$$\implies x = \frac{-\sigma^2 f''(x)}{f'(x)}$$

This observation tells us that there is a way to infer the position x along a boundary by just knowing σ, first and second directional derivatives along the gradient direction. In this way, we would have a way to map data value back to position along a boundary.

5 Contribution

The algorithm for transfer function generator is designed as a two-step process. The first step focuses on the computation of boundary characteristics of the input volume. In this step, the algorithm finds the data values corresponding to the object. In order to visualize different parts of the object, data values corresponding to the object are classified into multiple subsections using accumulative probability distribution of volume data. Then, a threshold data value is computed in each subsection. These threshold data values act as piecewise nodes of the opacity transfer function which is given to Direct Volume Rendering algorithm. The user can set opacity to each threshold data value using sliders and visualize the changes in the rendered volume instantly.

Taking this algorithm as basis, we have designed UML class diagram and sequence diagram to understand the interaction between classes in a better way. The algorithm has been implemented in C++ with the support of ITK[Insight Toolkit] and VTK[Visualization Toolkit] in visual studio. The C++ program takes 3D volume as input and generates a transfer function which is used by the direct volume rendering algorithm to produce the final rendered volume as output.

Process:

{

Input: 3D volume

Figure out the data values that correspond to boundaries.

Classification of data values corresponding to the object into sub sections.

Threshold data value generation in each sub section.

Set threshold data values as piecewise nodes of opacity transfer function.

Direct Volume Rendering

Output: Rendered Volume

User can change the opacity transfer function using sliders.

}

5.1 Algorithm

5.1.1 Computation of Boundary Characteristics

The algorithm is based on figuring out the data values corresponding to the overall boundary of the object using this information to guide the user towards opacity function settings which readily display the boundaries.

Input: 3D volume(CT Scan), histogram volume size.

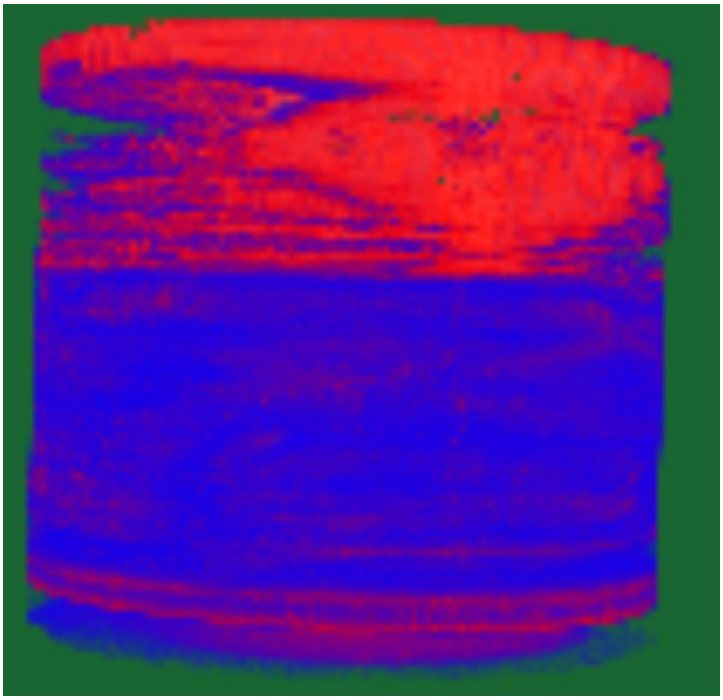


Figure 2: Input Volume: Savings Box

STEP1: Calculate the first and second directional derivatives along the gradient direction for the given input volume.

Reason: The direction of the gradient obviously varies throughout the volume, but the observed relationship between data values and its directional derivatives is constant, because the boundary is assumed uniform everywhere. At the mid-point of the boundary, the first derivative is at a maximum, and the second derivative has a zero-crossing.

STEP2: Transformation from Position Domain to Value Domain.

Reason: We need to find out the data values that correspond to boundaries. So we need to find the relationship between data values and its derivatives irrespective of the position. For every data value, find its corresponding gradient and second derivative magnitude.

STEP3: Histogram Volume Creation

Reason: We are interested in the probabilities associated with three variables i.e. data value, first derivative and second derivative. A histogram can give us a compact summary of large amount of data. With the three variables as the axes of a 3D histogram we could get the frequency for every possible combination of these three variables.

STEP4: Calculate Mean First Directional Derivative(g[v]) and Mean Second Directional Derivative(h[v]).

Reason: A data value can be present in various locations and at every location it might have a different first and second directional derivative magnitudes. In order to have a fixed first and second directional derivative magnitude for each data value, mean of these quantities is computed using weighted average.

$$g[v] = \frac{\sum_{i=1}^{n} g[i]f[i]}{\sum_{i=1}^{n} f[i]} \qquad\qquad h[v] = \frac{\sum_{i=1}^{n} h[i]f[i]}{\sum_{i=1}^{n} f[i]}$$

where v is the data value, i represents a particular index in the histogram volume and n is the size of histogram volume.

f[i] = frequency at index i in the histogram volume.

g[i] = first directional derivative magnitude at index i.

h[i] = second directional derivative magnitude at index i.

STEP5: Generate Position Function(p[v])

Reason: Based on the boundary model presented by Kindlmann and Durkin[4], the second directional derivative attains its extrema at $\pm\sigma$ (where σ is the usual standard deviation). Then thickness of the boundary is defined to be 2σ. Knowing g[v] and h[v], we can calculate σ and position function(p[v]).

$$\sigma = \frac{2\sqrt{e}(max_v(g[v]))}{max_v(h[v]) - min_v(h[v])}$$
$$p[v] = \frac{-\sigma^2 h[v]}{g[v]}$$

Position function(p[v]) tells us the distance of a data value from the midpoint of the boundary. If h[v] is minimum and g[v] is maximum, then that data value corresponds to boundary.

STEP6: Generate Opacity Function($\alpha[v]$)

Reason: Thickness of the boundary is defined to be 2σ.

$$\text{For a data value v, if } -\sigma \leq p[v] \leq +\sigma \Rightarrow \text{ valid data value}$$

The opacity of every valid data value is set to the opacity specified by the user.

STEP7: Give the opacity function to Direct Volume Rendering algorithm.

Output: Rendered Volume

Figure 3: Rendered Volume of Savings Box after performing the above steps.

The above steps helped us to find the overall boundary of the object in the given input volume but the user cannot visualize different parts of the object and same opacity is assigned to all the valid data values. The second phase of the algorithm helps in solving all these problems.

5.1.2 Classification of Volume Data using Minimum Cross Entropy Thresholding(MCET)

Input: 3D volume(CT Scan)
Here, the input 3D volume is the output of Computed Tomography reconstruction algorithm and the data value is a 16 bit grey scale.
Step1: Compute the frequency of each data value.
Let (x,y,z) be the position of the voxel in the input volume and v(x,y,z) be the data value at that position.

$$\text{if } v(x,y,z) = i \quad \text{where } i \in [0,65535]$$
$$\text{frequency}(i) = \text{frequency}(i) + 1$$

STEP2: Construction of Probability Distribution of Volume Data.
Let the dimensions of the input volume be (p,q,r).

$$\text{d(i)} = \frac{frequency(i)}{p*q*r}$$

where d(i) is the probability distribution of volume data.
STEP3: Partition of probability distribution of volume data.
Assume there are m classes in the volume data, then probability distribution of volume data is partitioned into m subsections by m-1 points.

- Firstly the nonzero endpoint of probability distribution v_{min} and v_{max} are determined.
- The data values in the range $v_{min} \sim v_{max}$ are segmented into m parts averagely and endpoint of each subsection is computed.
- The accumulative probability distribution of each subsection is computed.

  $$d_s = \sum_{i=v_s}^{v_{s+1}} d(i)$$

  where v_s and v_{s+1} are the end points of subsection s.
  d_s is the accumulative probability distribution of subsection s.
- If the accumulative probability distribution of a subsection is very small(i.e. less than 1/m), then the number of subsections is reduced and range of that subsection is increased. So that each subsection contributes equally to the input volume.

This step splits the probability distribution of volume data into some subsections. Now, a threshold data value is to be computed in each subsection using minimum cross entropy.
STEP4: Calculation of threshold data values. Let s be a subsection with v_s and v_{s+1} as its endpoints. Let t be the threshold data value of this subsection such that $v_s < \text{t} < v_{s+1}$. The cross entropy is then calculated by

$$\text{D(t)} = \sum_{i=v_s}^{t-1} id(i) log(\frac{i}{\mu(v_s,t}) + \sum_{i=t}^{v_{s+1}} id(i) log(\frac{i}{\mu(t,v_{s+1}+1)})$$
$$\text{where } \mu(a,b) = \frac{\sum_{i=a}^{b-1} id(i)}{\sum_{i=a}^{b-1} d(i)}$$

The MCET determines the optimal threshold t^* for a subsection by minimizing the cross entropy[9] i.e.

$$t^* = min_t(\text{D(t)})$$

Let the range of subsection be r i.e. r = v_{s+1} - v_s. Then, the computational complexity of determining optimal threshold in this subsection is O(r^2). This time complexity can be reduced to constant time using recursive programming[10].

Recursive Programming

The MCET objective function can be rewritten as:

$$\begin{aligned} \mathrm{D(t)} &= \sum_{i=v_s}^{v_{s+1}} id(i)log(i) - \sum_{i=v_s}^{t-1} id(i)log(\mu(v_s,t)) - \sum_{i=t}^{v_{s+1}} id(i)log(\mu(t,v_{s+1}+1)) \\ &= \sum_{i=v_s}^{v_{s+1}} id(i)log(i) - (\sum_{i=v_s}^{t-1} id(i))log(\frac{\sum_{i=v_s}^{t-1} id(i)}{\sum_{i=v_s}^{t-1} d(i)}) - (\sum_{i=t}^{v_{s+1}} id(i))log(\frac{\sum_{i=t}^{v_{s+1}} id(i)}{\sum_{i=t}^{v_{s+1}} d(i)}) \\ &= \sum_{i=v_s}^{v_{s+1}} id(i)log(i) - m^1(v_s,t)log(\frac{m^1(v_s,t)}{m^0(v_s,t)}) - m^1(t,v_{s+1}+1)log(\frac{m^1(t,v_{s+1}+1)}{m^0(t,v_{s+1}+1)}) \end{aligned}$$

where $m^0(a,b) = \sum_{i=a}^{b-1} d(i)$ and $m^1 = \sum_{i=a}^{b-1} id(i)$ are the zero-moment and first-moment on partial range of probability distribution of volume data.

If D(t) is calculated for one t in one subsection, then for all other t's in the same subsection, D(t) can be calculated in constant time. After computing $m^0(v_s,t)$, $m^1(v_s,t)$, $m^0(t,v_{s+1}+1)$ and $m^1(t,v_{s+1}+1)$, the other moments can be computed recursively by the following equation:

$$m^0(v_s,t+1) = m^0(v_s,t) + d(t) \qquad m^0(t+1,v_{s+1}+1) = m^0(t,v_{s+1}+1) - d(t)$$
$$m^1(v_s,t+1) = m^1(v_s,t) + td(t) \qquad m^1(t+1,v_{s+1}+1) = m^1(t,v_{s+1}+1) - td(t)$$

By, this method we can compute thresholds in each subsection in constant time.

STEP5: The position function(p[v]) of the threshold data values in each subsection is checked in the following manner:

For a threshold data value t, if $-\sigma \leq p[t] \leq +\sigma$ => valid threshold data value

The valid threshold data values calculated according to the algorithm are united as the piecewise nodes of the opacity transfer function.

STEP6: Give the opacity transfer function to Direct Volume Rendering algorithm.

Assign default opacity to each piecewise node of the transfer function and give it to DVR algorithm.

STEP7: Rendered Volume

The initial rendered volume has the default opacity. But one slider is provided for each threshold data value and the user has the provision to change the opacity of that threshold data value. The changes made by the user in opacity transfer function with the help of sliders are applied instantly to the rendered volume.

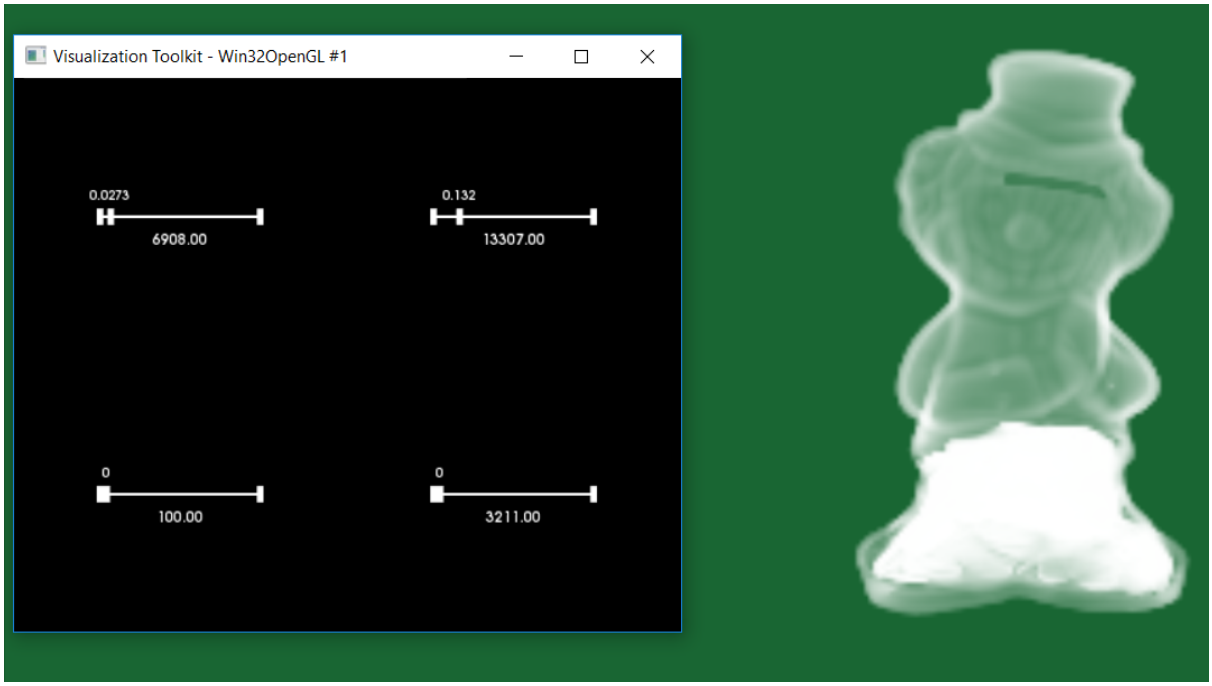


Figure 4: Final Rendered Volume of Savings Box with user interactive sliders to modify opacity transfer function.

In this rendered volume, we could see the coins inside the savings box as well as the overall boundary of the savings box.

6 Implementation

The program is entirely written in C++ and has been developed exploiting Object-Oriented methodologies and tools. The C++ classes in this program use templates in order to accept inputs of different types from the user. The program is implemented with the support of ITK[Insight Toolkit] and VTK[Visualization Toolkit] on visual studio. The working of the program can be understood with the help of flowcharts.

- Main Module: This module acts as starting point for the program execution. All the objects of user defined classes are created here and the user has to set two input parameters: input file name and histogram volume size.

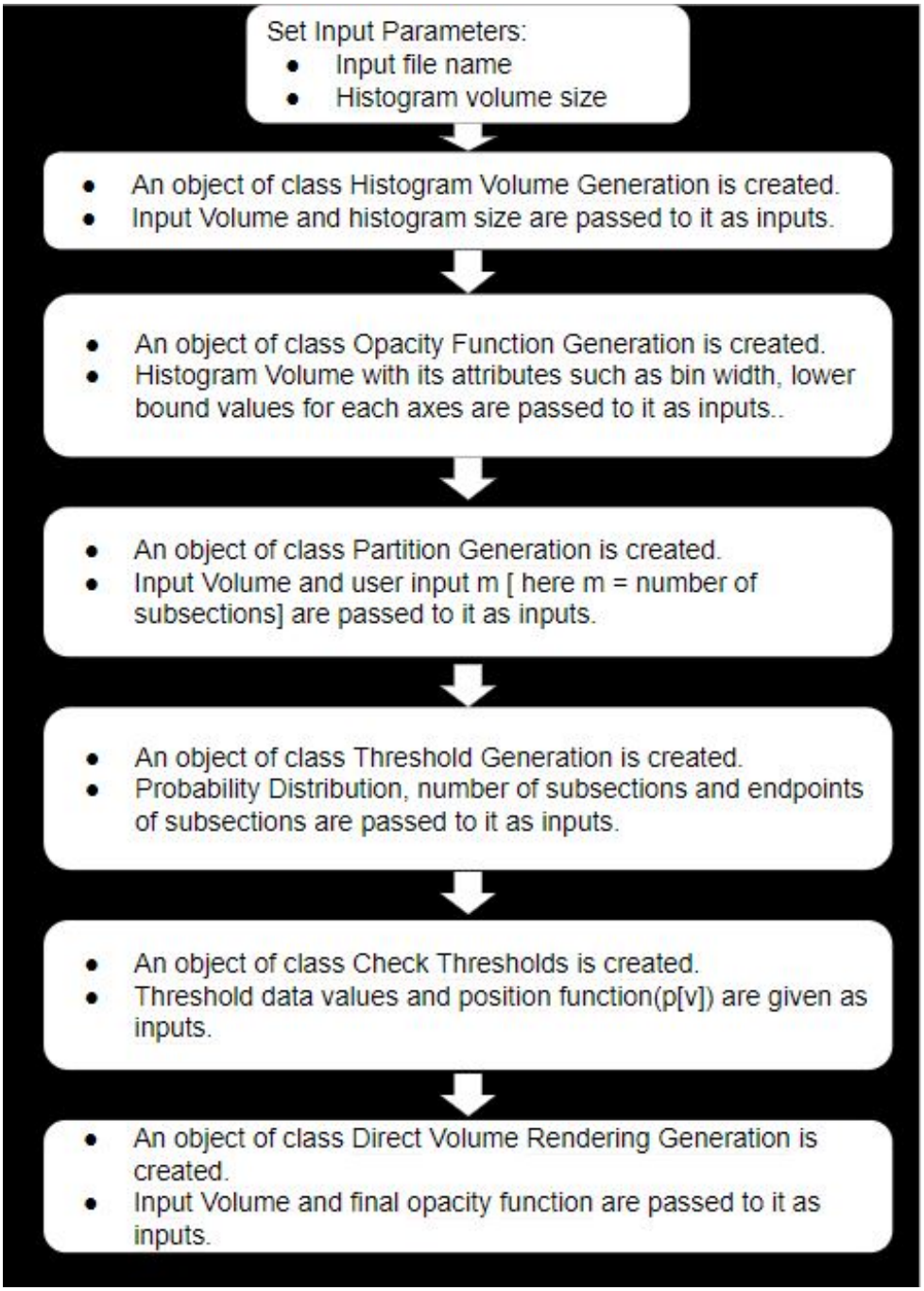


Figure 5: Flowchart of main.cpp

- Histogram Volume Generation Module: In this module, 3D histogram is created with data value, gradient value and second derivative value as X, Y and Z axes respectively. Each bin of 3D histogram stores the frequency of combination of data value, gradient

value and second derivative value at that index.

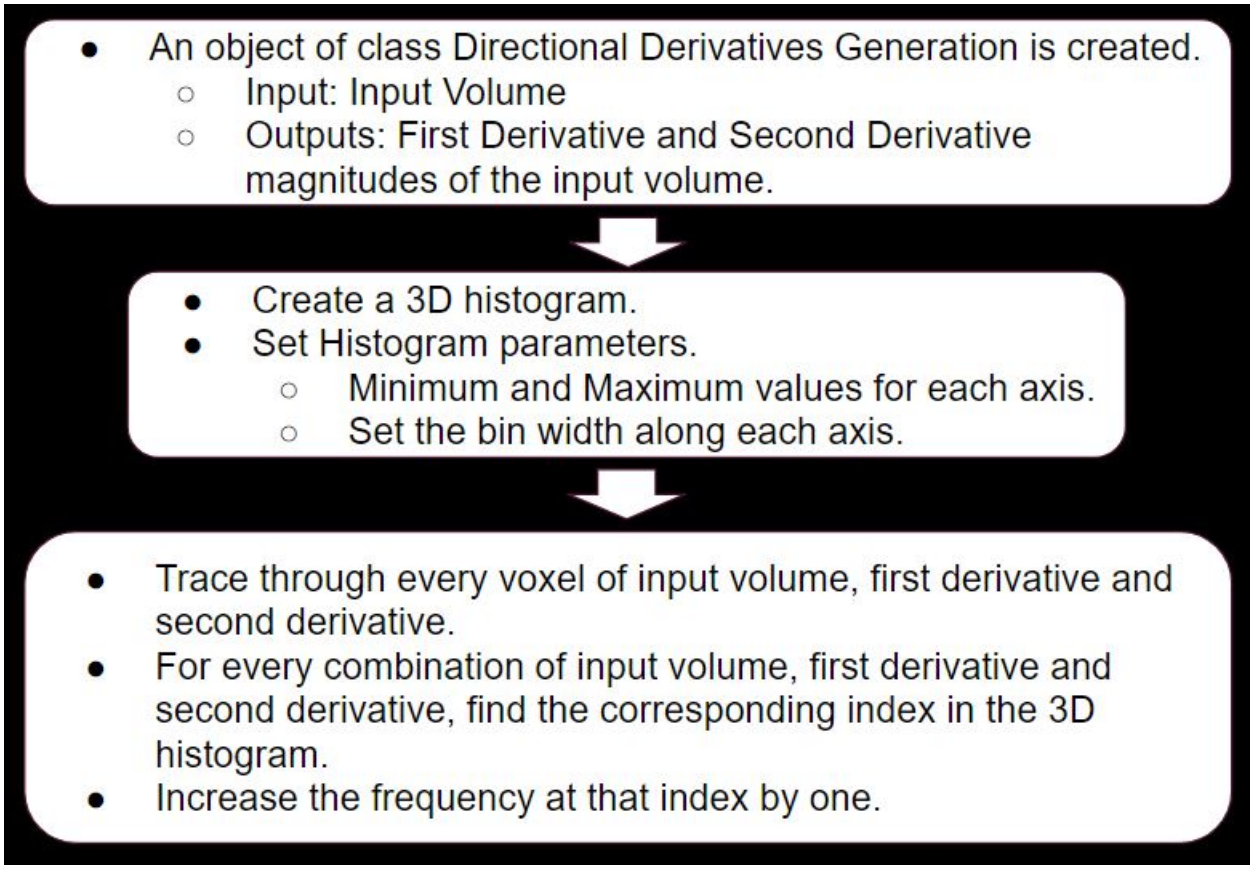


Figure 6: Flowchart of HistogramVolumeGeneration.hxx

- Opacity Function Generation Module: Using the histogram, the average first directional derivative ($g[v]$) and average second directional derivative ($h[v]$) are calculated for each data value . $g[v]$ and $h[v]$ are used to compute σ and position function ($p[v]$). All the data values with p[v] in the range -σ to +σ are valid data values.

- Final Opacity Function Generation Module: The input volume can have different objects but identifying each object and assigning it a different opacity is a difficult task. As a first step, we tried to differentiate between the data values that correspond to boundaries of the objects and other that do not belong to boundaries. After grouping the data values into these two categories, different opacity is assigned to each category.

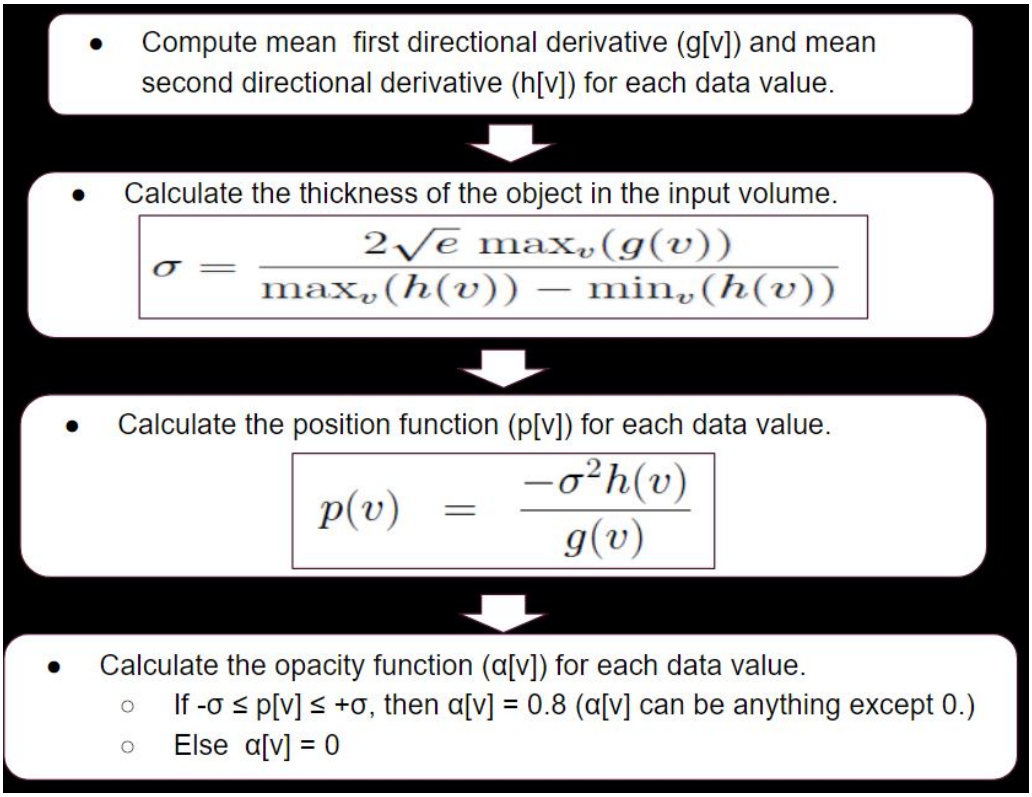


Figure 7: Flowchart of OpacityFunctionGeneration.hxx

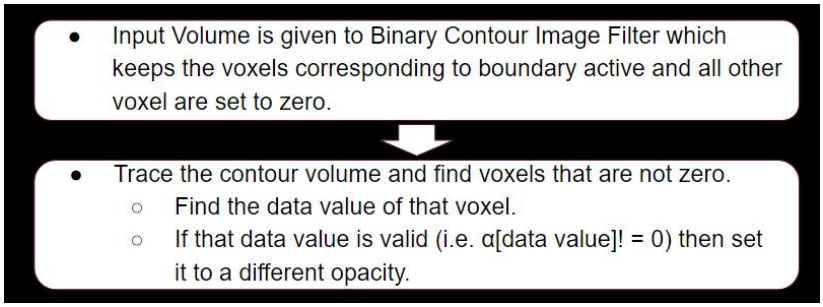


Figure 8: Flowchart of FinalOpacityFunctionGeneration.hxx

- Partition Generation Module: The input volume is split into subsections using accumulative probability distribution.
- Threshold Generation Module: The threshold data value for each subsection is computed using minimum cross entropy thresholding algorithm. The process of finding thresholds is made fast using recursive programming.

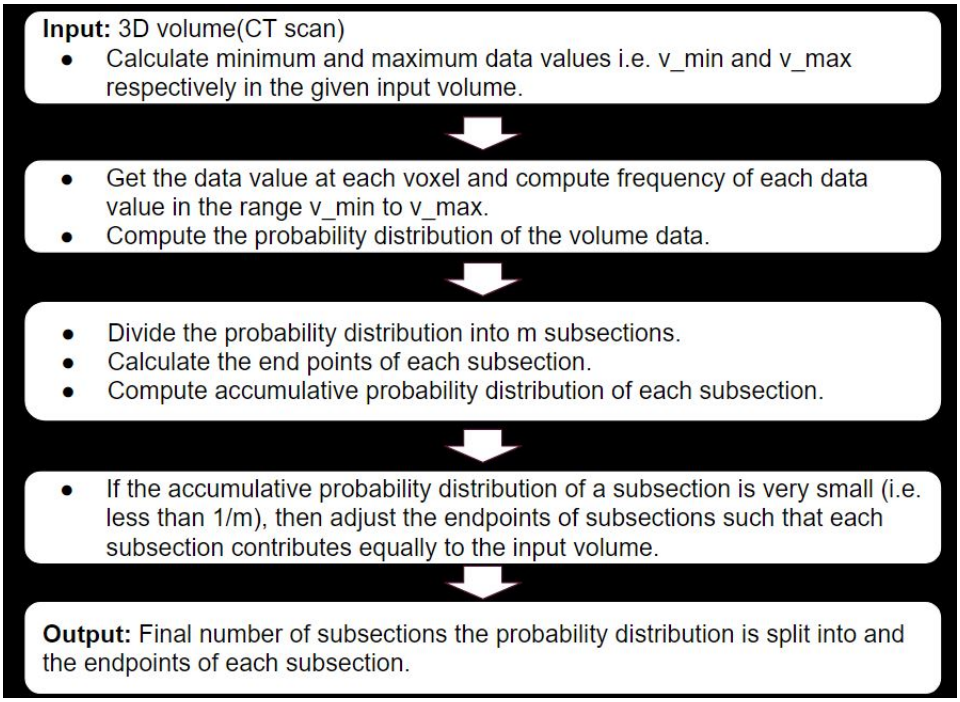


Figure 9: Flowchart of PartitionGeneration.hxx

- Check Threshold Module: The threshold data values whose position function(p[v]) is between -σ to +σ are valid threshold data values.

- Direct Volume Rendering Module: The opacity transfer function is usually designed as piecewise functions for flexibly displaying each section of the whole object. The valid threshold data values act as piecewise nodes and a default opacity is assigned to each of these piecewise nodes. The input volume is rendered using this opacity transfer function.

- User Interaction: User interactive sliders are provided in order to allow the users to set the opacity for the input volume. One slider is displayed for each valid threshold data value. The user can move the slider and set the opacity of corresponding threshold data value between [0,1]. The changes made by the user are instantly applied to the rendered volume.

7 Results

We have tested the program with the following inputs and the results are analysed manually.

7.1 Shepp-Logan Phantom

Dimensions of the input volume: 128 * 128 * 128.
Dimensions of the histogram volume: 256 * 256 * 256.

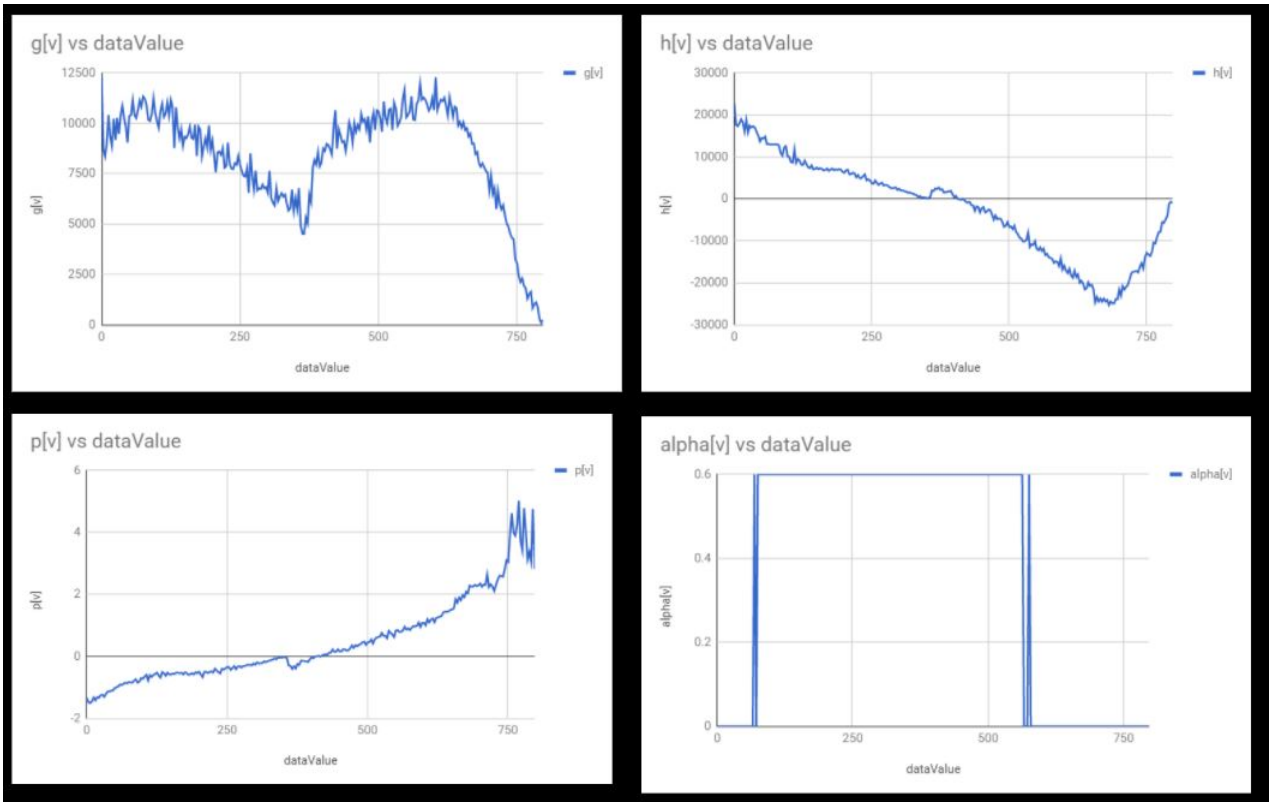


Figure 10: Relationship of data value with g[v], h[v], p[v] and $\alpha[v]$

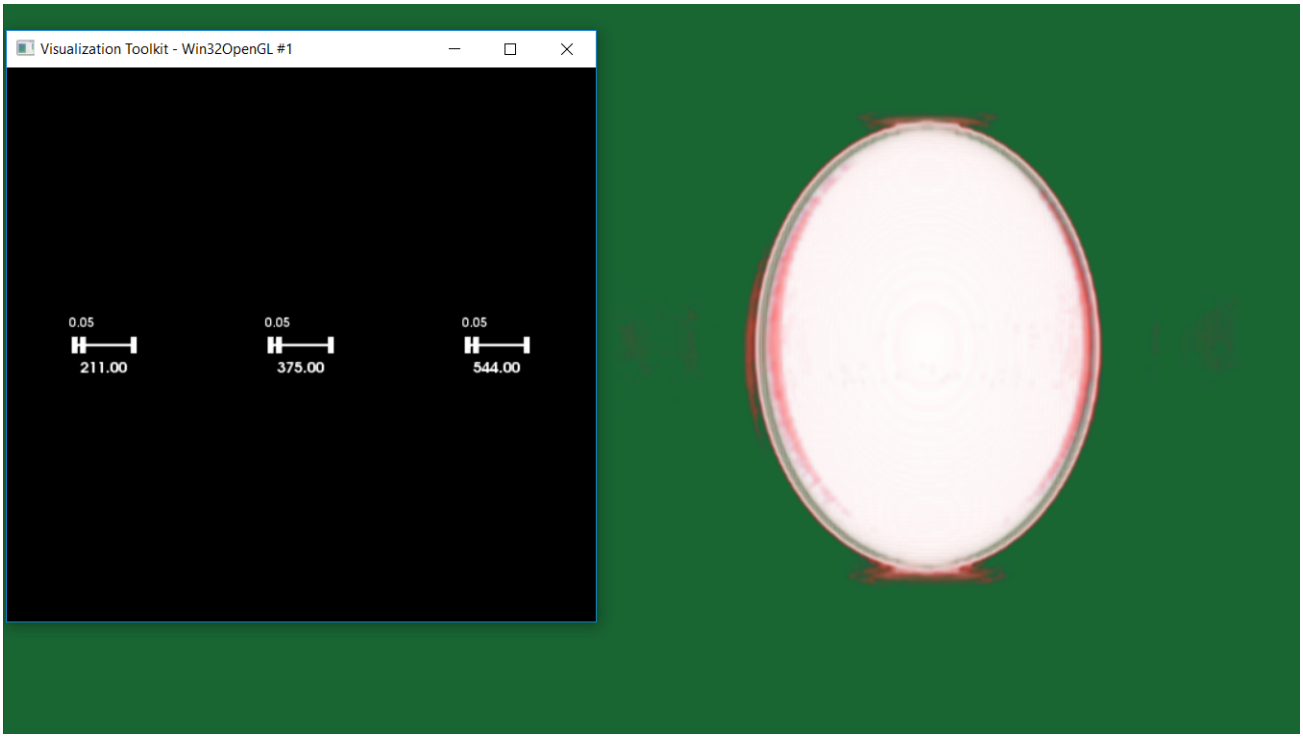


Figure 11: Output: Rendered Volume with Default opacity.

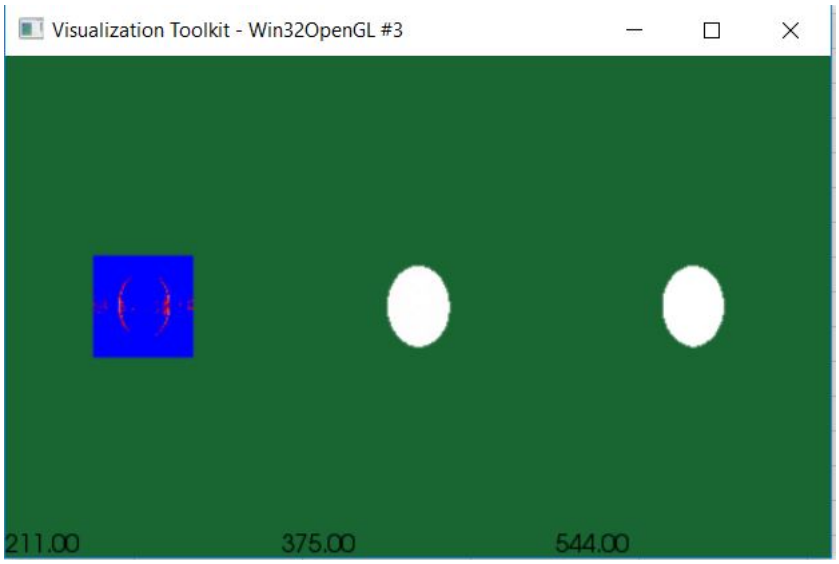


Figure 12: One rendering for each threshold data value to understand its contribution in the total input volume.

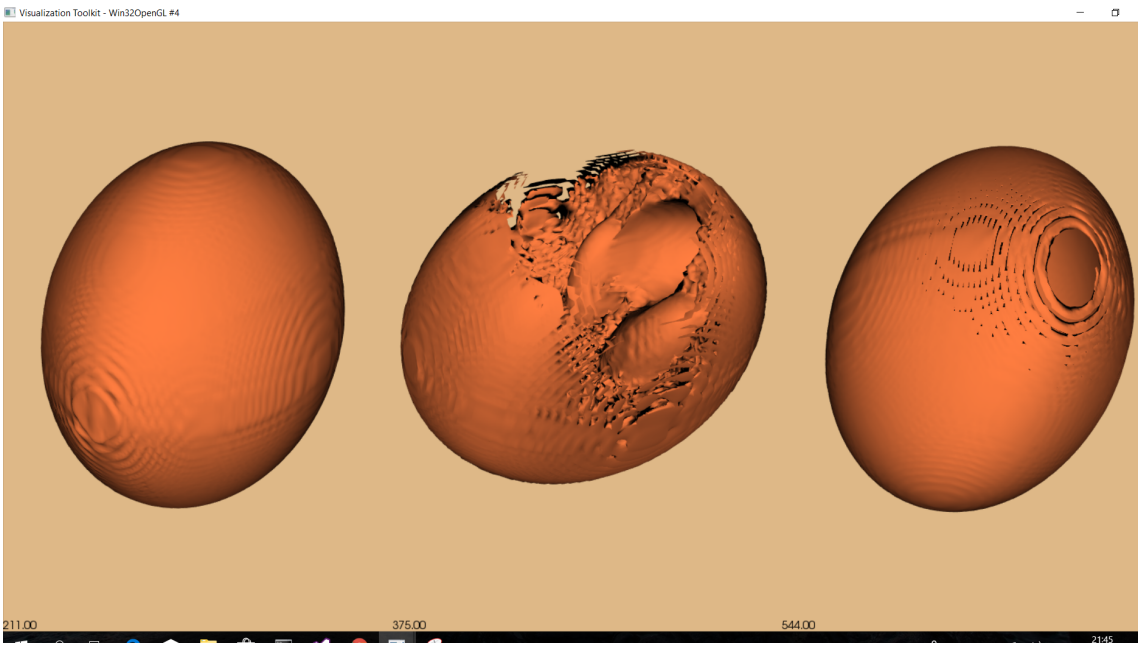


Figure 13: Iso-surface extraction of each threshold data value.

Figure 14: Output: Final rendered volume after applying changes made to the opacity transfer function by the user.

7.2 Papaya

This input volume has a papaya with seeds inside it. The papaya is placed on a papaya holder and then its CT scan is taken.
Dimensions of the input volume: 600 * 1000 * 600.
Dimensions of the histogram volume: 256 * 256 * 256.

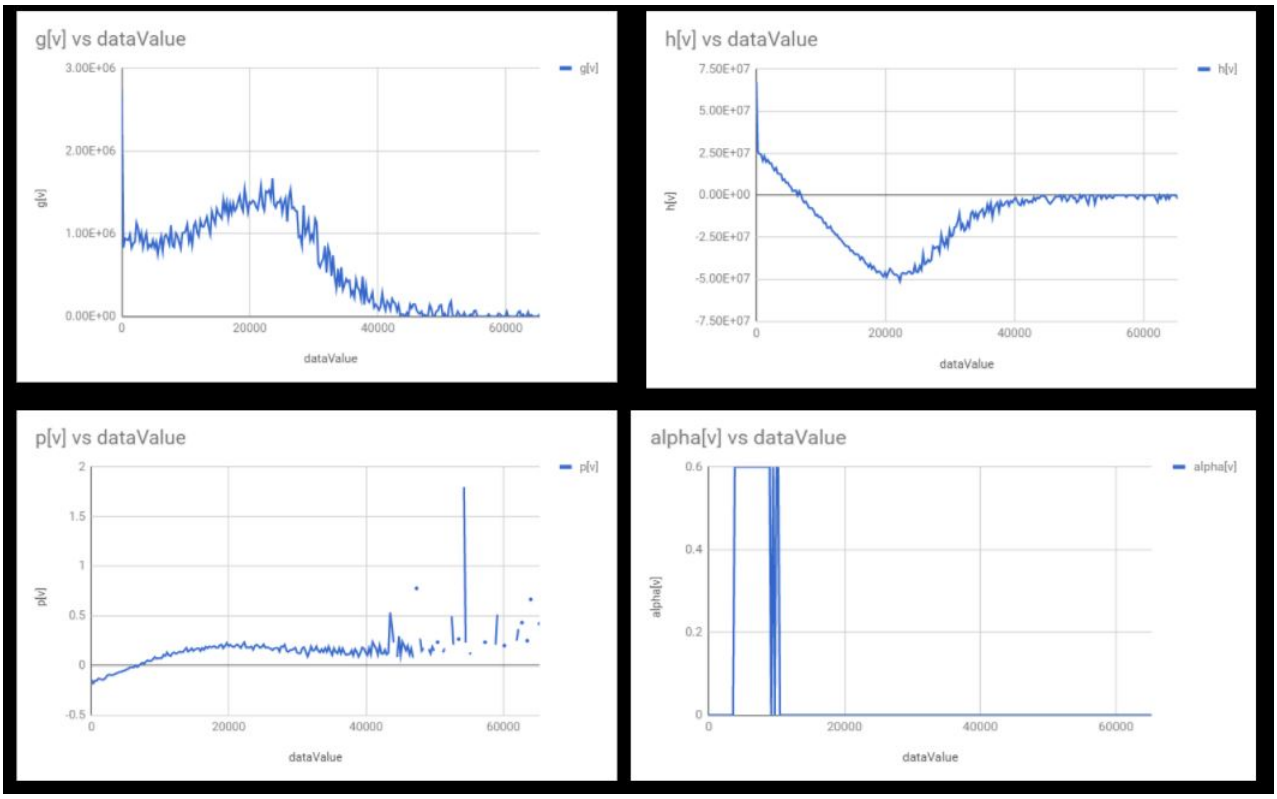


Figure 15: Relationship of data value with g[v], h[v], p[v] and $\alpha[v]$.

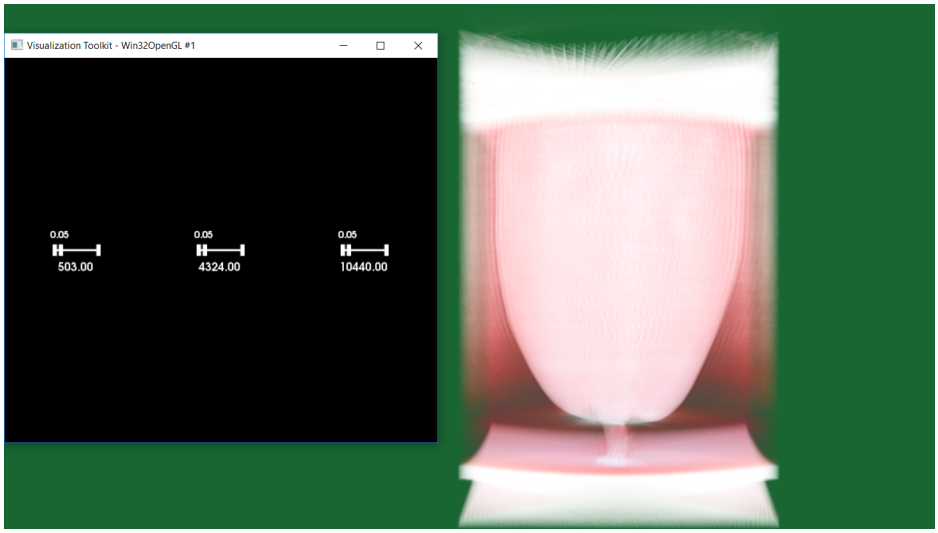


Figure 16: Output: Rendered Volume with Default opacity.

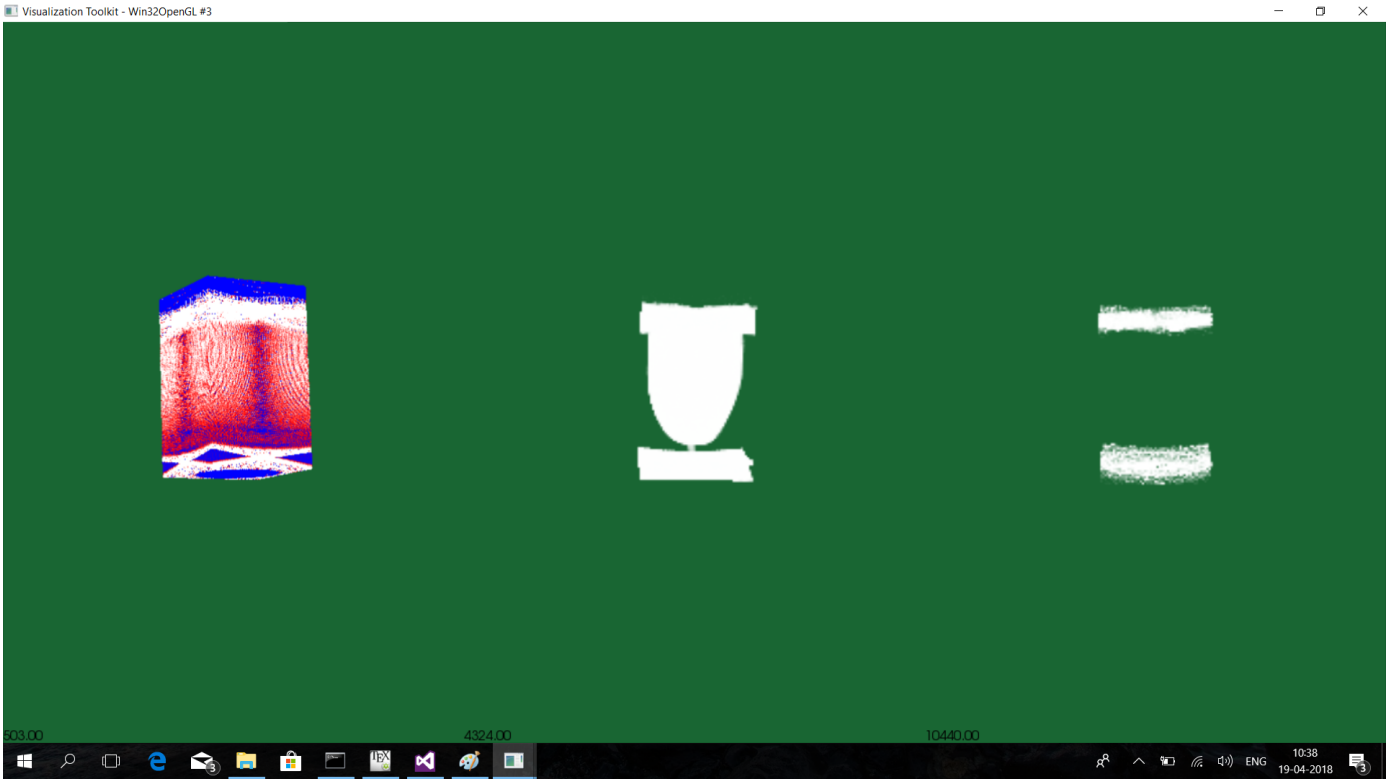


Figure 17: Output: One rendering for each threshold data value to understand its contribution in the total input volume.

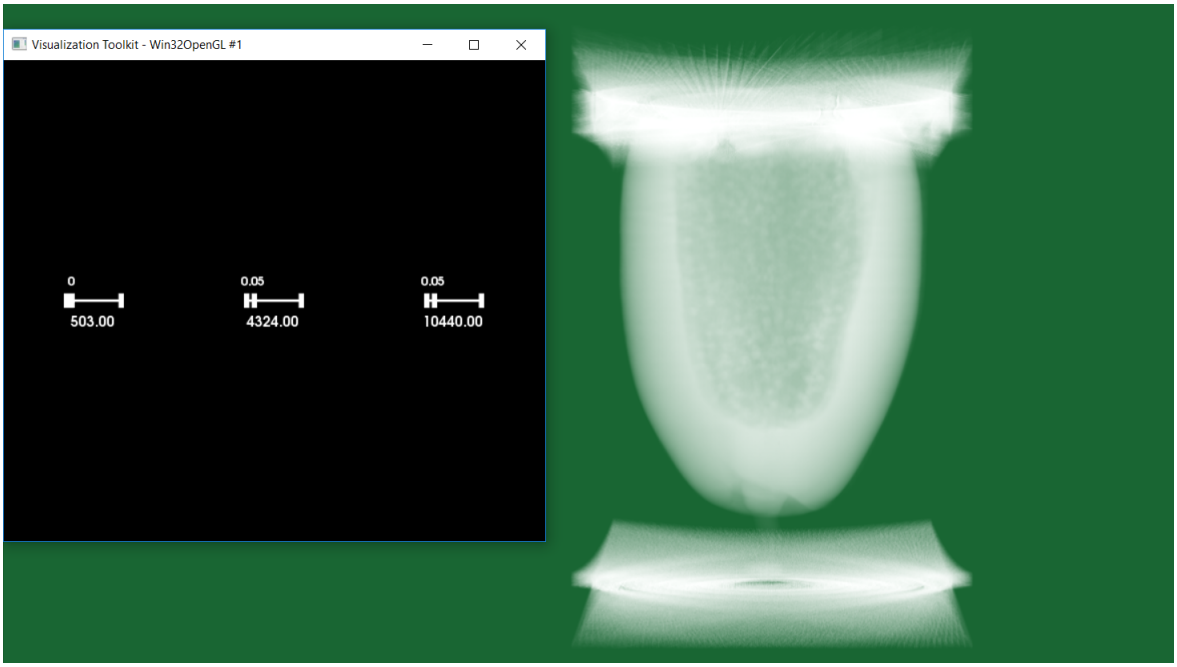


Figure 18: Output: Final rendered volume after applying changes made to the opacity transfer function by the user.

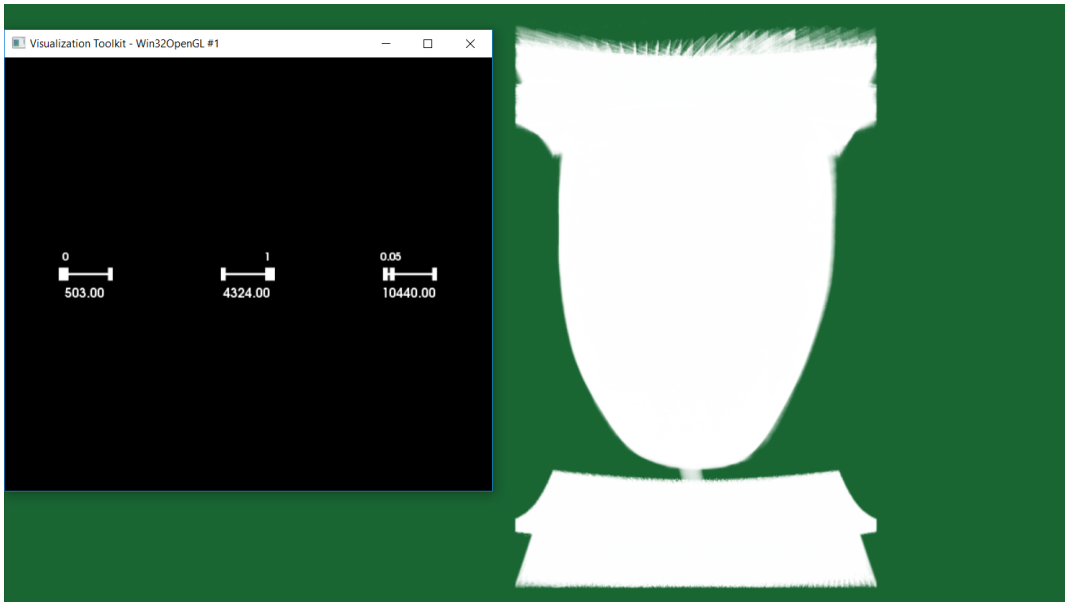
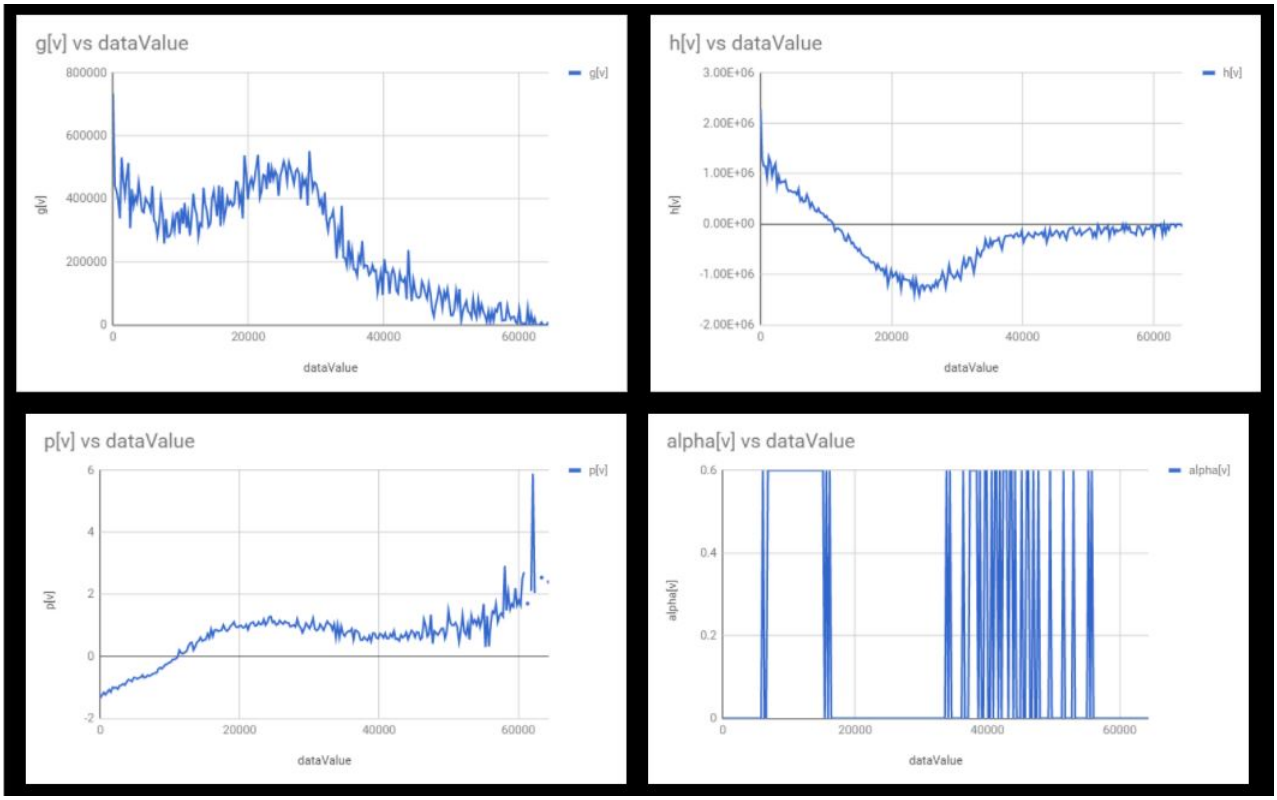


Figure 19: Output: Rendered volume after applying changes made to the opacity transfer function by the user.

7.3 Savings Box

This input volume has a savings box with the coins inside it. To see the coins inside the savings box, the data values corresponding to the boundary of the savings box and the data values that represent the coins must be given different opacities.
Dimensions of the input volume: 511 * 511 * 460.
Dimensions of the histogram volume: 256 * 256 * 256.



Relationship of data value with g[v], h[v], p[v] and $\alpha[v]$.

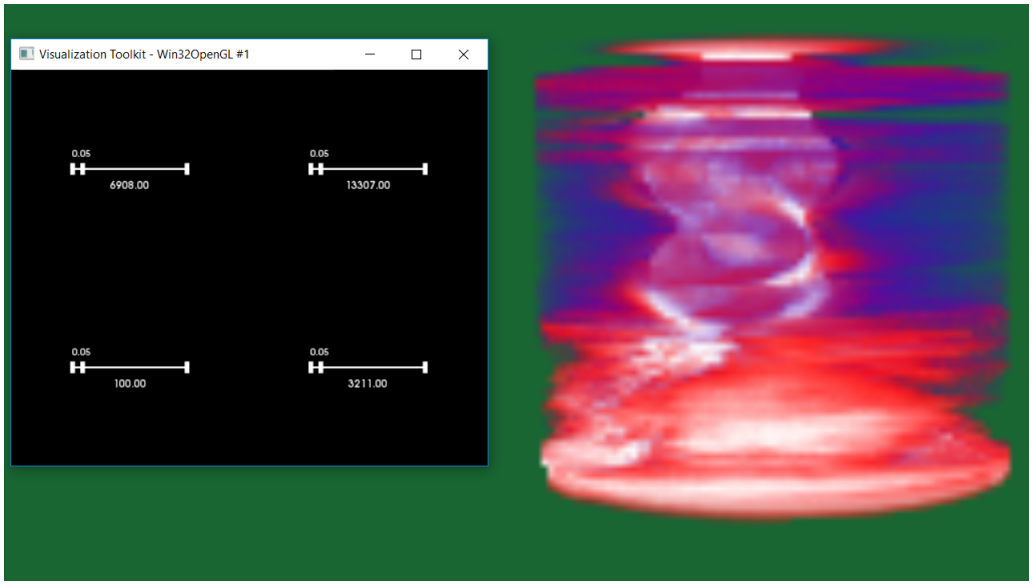


Figure 20: Output: Rendered Volume with Default opacity.

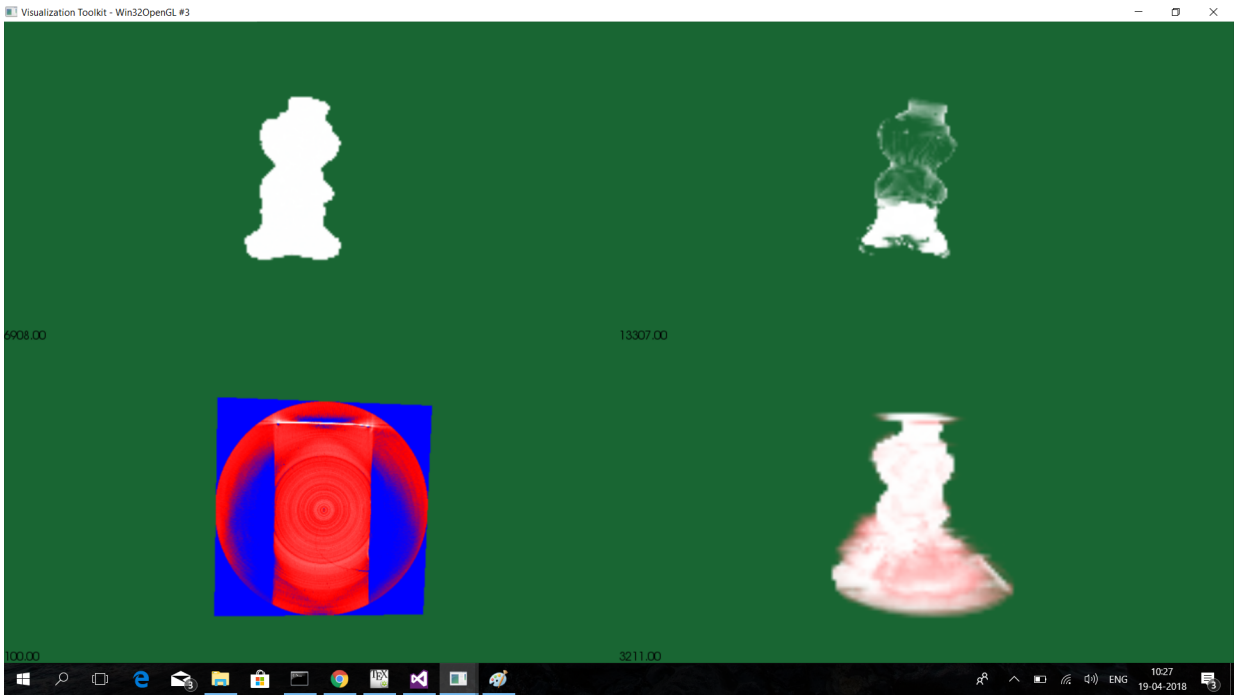


Figure 21: Output: One rendering for each threshold data value to understand its contribution in the total input volume.

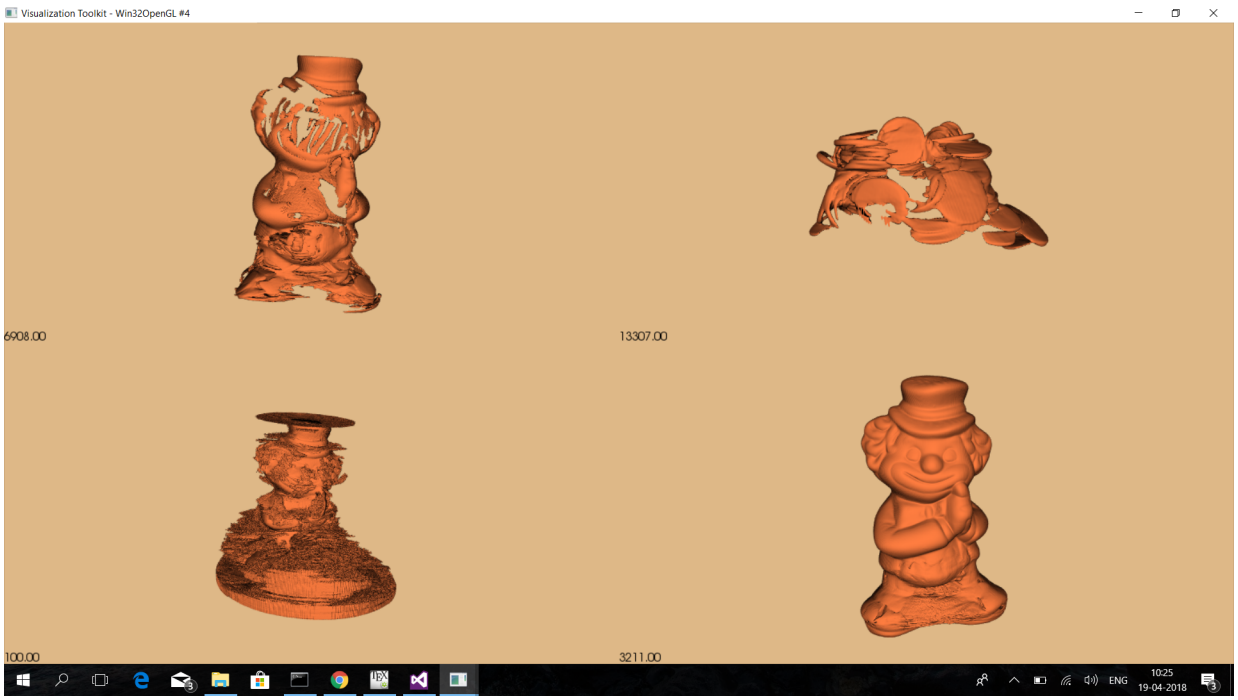


Figure 22: Output: Iso-surface extraction of each threshold data value.

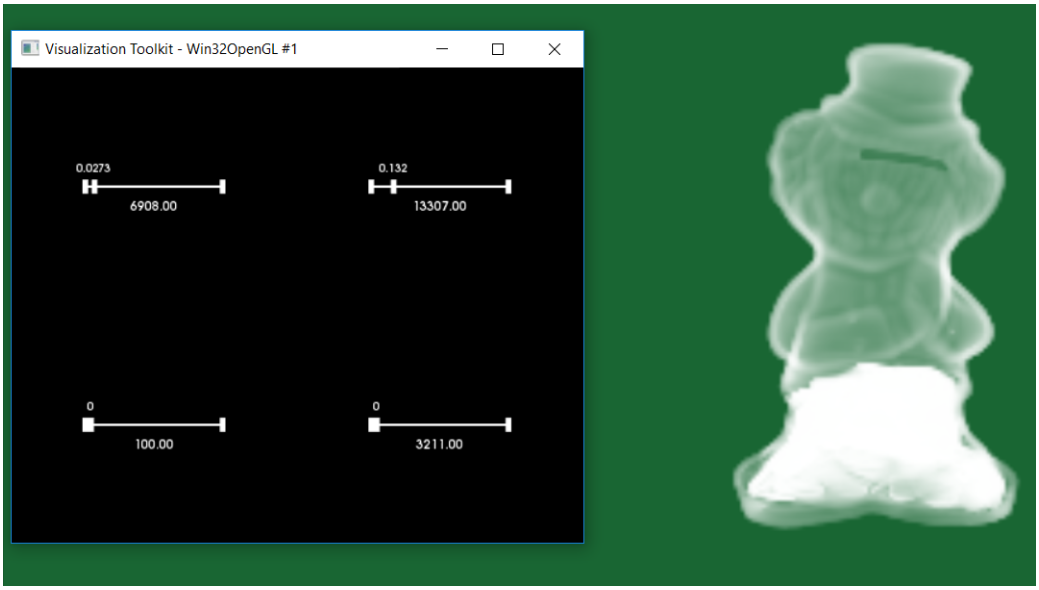


Figure 23: Output: Final rendered volume after applying changes made to the opacity transfer function by the user.

The main reasons to choose the above inputs to test our program are:

- Shepp-Logan Phantom is taken as input to confirm that the algorithm is correct and the program is rendering the given input volume.
- The savings box input volume has two main components: savings box which is made out of plastic and coins which are metal alloys. The components are made up of different materials which means they have different densities. The algorithm is able to identify the components with different densities.
- The program also allows the user to view the coins inside the coin box clearly.
- The savings box has a complex geometry and the algorithm is able to render the surface of the savings box clearly even after that.
- The papaya volume is taken as input as it is known to everyone. The seeds in papaya are much smaller when compared to its outer covering but the algorithm is able to bring these seeds into focus. The internal features of papaya i.e. seeds are clearly made visible to the user.

In this way, the above three inputs helps us to understand the power of this algorithm.

8 Conclusion

A transfer function given to direct volume rendering algorithm plays an important role in the process of visualization. But setting a good transfer function for a particular input volume is a difficult task. In this project, the algorithm for transfer function generator is designed as a two-step process. The first step focuses on the computation of boundary characteristics of the input volume. And in the second step, the data values corresponding to the object are segmented into subsections and a threshold data value is computed for each subsection using minimum cross entropy. A rendering is produced for each threshold data value to make the user understand the contribution of that data value to the whole input volume. With this knowledge, the user can set opacity to each threshold data value using sliders and visualize the changes in the rendered volume instantly. We have built a transfer function generator that works at interactive speed and also displays internals features of an object with minimum user interaction.

9 Future Work

Some interesting areas to explore are:

- Automatic detection of objects within complex CT volume: The automatic detection of objects within complex volumetric imagery is becoming of increased interest due to the use of dual energy Computed Tomography (CT) scanners as an aviation security deterrent.

- Real-time volumetric deformation: The ability to accurately model local deformation is extremely important in medicinal application such as minimal invasive surgery and computer assisted intervention. In a typical clinical application scenario, tomography data is acquired before the intervention for a detailed surgery planning. During the intervention however the pre-operatively acquired image data does not match the actual situation due to anatomical shifts and tissue resection. In consequence the spacial misalignment must be compensated by adapting the volume data to the non-linear distortion.

- Finding the physical dimensions of the features in the input volume: Most volume rendering research is established on a unit scale of voxels, and not physical dimensions. Therefore, the result depends on the underlying sampling density. As such, it leaves most relevant measures out of touch with end users and domain applications, making it difficult to reproduce. For example, finding a feature in a medical dataset, whose size is given like "three voxels wide", is entirely irrelevant. Medical features should have width expressed in a physical unit relating to human physiology.

References

[1] J. Marks, B. Andalman, and P. A. Beardsley, "Design galleries: A general approach to setting parameters for computer graphics and animation," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '97, 1997, pp. 389–400. [Online]. Available: http://dx.doi.org/10.1145/258734.258887

[2] T. He, L. Hong, A. Kaufman, and H. Pfister, "Generation of transfer functions with stochastic search techniques," in *Visualization '96. Proceedings.*, Oct 1996, pp. 227–234.

[3] S. Fang, T. Biddlecome, and M. Tuceryan, "Image-based transfer function design for data exploration in volume visualization," in *Visualization '98. Proceedings*, Oct 1998, pp. 319–326.

[4] G. Kindlmann and J. W. Durkin, "Semi-automatic generation of transfer functions for direct volume rendering," in *IEEE Symposium on Volume Visualization (Cat. No.989EX300)*, Oct 1998, pp. 79–86.

[5] J.-S. Prani, T. Ropinski, and K. Hinrichs, "Efficient boundary detection and transfer function generation in direct volume rendering," in *Proceedings of the Vision, Modeling, and Visualization Workshop 2009*, Jan 2009, pp. 285–294.

[6] F.-Y. Tzeng and K.-L. Ma, "A cluster-space visual interface for arbitrary dimensional classification of volume data," in *In Proceedings of Joint Eurographics-IEEE TVCG Symposium on Visualization*, May 2004, pp. 17–24.

[7] J. JaJa, A. Varshney, and C. Y. Ip, "Hierarchical exploration of volumes using multilevel segmentation of the intensity-gradient histograms," *IEEE Transactions on Visualization Computer Graphics*, vol. 18, pp. 2355–2363, 12 2012. [Online]. Available: doi.ieeecomputersociety.org/10.1109/TVCG.2012.231

[8] L. Wang, X. Chen, S. Li, and X. Cai, "General adaptive transfer functions design for volume rendering by using neural networks," in *Neural Information Processing, 13th International Conference, ICONIP 2006*, vol. 18, Aug 2006, pp. 661–670.

[9] J. Li, L. Zhou, H. Yu, H. Liang, and L. Zhang, "Classification for volume rendering of industrial ct based on minimum cross entropy," in *2007 International Conference on Mechatronics and Automation*, Aug 2007, pp. 2710–2715.

[10] Y. Zhao, Z. Fang, K. Wang, and H. Pang, "Multilevel minimum cross entropy threshold selection based on quantum particle swarm optimization," in *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, vol. 2, July 2007, pp. 65–69.

[11] P. Ljung, J. Krüger, E. Gröller, M. Hadwiger, C. D. Hansen, and A. Ynnerman, "State of the art in transfer functions for direct volume rendering," in *Proceedings of the Eurographics / IEEE VGTC Conference on Visualization: State of the Art Reports*, ser. EuroVis '16, 2016, pp. 669–691. [Online]. Available: https://doi.org/10.1111/cgf.12934

[12] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. S. Avila, K. M. Raghu, R. Machiraju, and J. Lee, "The transfer function bake-off," *IEEE Computer Graphics and Applications*, vol. 21, no. 3, pp. 16–22, May 2001.

[13] Y. Wu and H. Qu, "Interactive transfer function design based on editing direct volume rendered images," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 5, pp. 1027–1040, Sept 2007.

[14] C. R. Salama, M. Keller, and P. Kohlmann, "High-level user interfaces for transfer function design with semantics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1021–1028, Sept 2006.

[15] H. C. Wong, U. H. Wong, and Z. Tang, "Direct volume rendering by transfer function morphing," in *2009 7th International Conference on Information, Communications and Signal Processing (ICICS)*, Dec 2009, pp. 1–4.

[16] T. Pfaffelmoser, M. Reitinger, and R. Westermann, "Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields," *Computer Graphics Forum*, vol. 30, no. 3, pp. 951–960, June 2011.