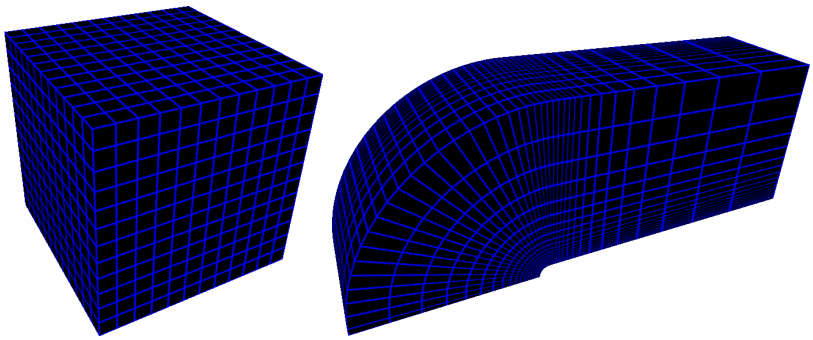
1
Introduction

Volume rendering is a popular way to visualize scalar fields from a number of different data, from medical MRI to results of scientific simulations. Unlike regular surface rendering, volume rendering allows the user to visualize the interior of the dataset. By computing the light intensity along rays as they traverse the volume, it is possible to calculate the color and opacity for the pixels on the screen. How to accurately calculate the interaction between light and volume, while maintaining an acceptable rendering time, is one of the main difficulties of volume visualization algorithms.

Important engineering applications use unstructured hexahedral meshes for numerical simulations. Hexahedral cells, when compared to tetrahedral ones, tend to be more numerically stable and to require less mesh refinement. However, volume visualization of unstructured hexahedral meshes is challenging due to the trilinear variation of scalar fields inside the cells. The conventional solution consists in dividing each hexahedron cell into five or six tetrahedra, approximating a trilinear variation by a piecewise linear function. This results in a less smooth variation, inaccurate images and increases the memory consumption.

Figures 1.1(a) and 1.1(b) present two typical types of meshes. The first one is an example of structured data. Such data is represented by a set of uniform size cells and, as such, its adjacency information can be implicitly retrieved (e.g. cell (i, j, k) is adjacent to cell $(i+1, j, k)$). Figure 1.1(b), on the other hand, is an unstructured data; its cells do not necessarily have an uniform size and its adjacency information can no longer be retrieved implicitly; we must store such information in a data structure.



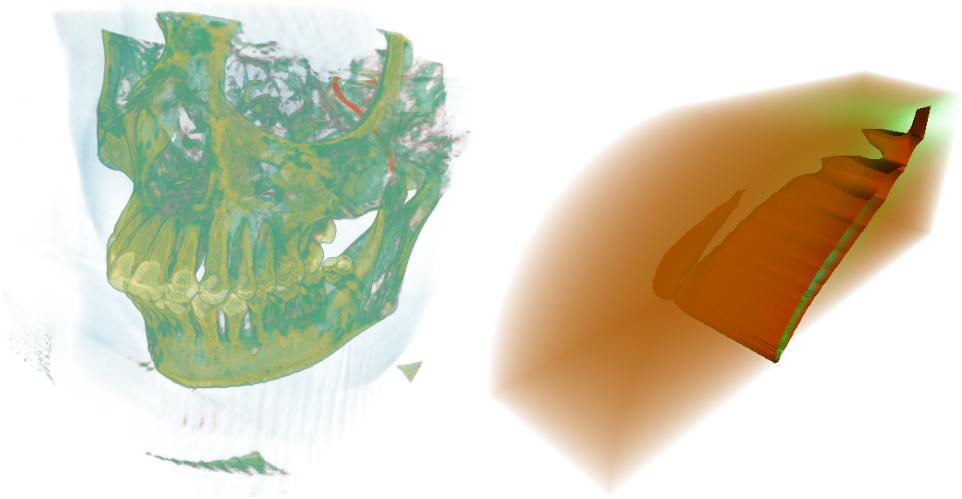
1.1(a): Structured regular mesh

1.1(b): Unstructured mesh

Figure 1.1: Meshes types.

Figures 1.2(a) and 1.2(b) present two typical applications of volume

visualization. The first one, an x-ray image, is a structured regular data, while Figure 1.2(b) illustrates an unstructured data of a computational fluid dynamics simulation.



1.2(a): X-Ray 1.2(b): Computational fluid dynamics

Figure 1.2: Volume rendering examples.[1]

One popular approach to render volumetric data, both structured and unstructured, is ray-casting (4). By tracing a ray for each pixel on the screen, we can traverse the volume and calculate the contribution of a set of discrete volume cells to the final pixel color, using an emission-absorption optical model (17). Because volumetric data are nothing more than a series of scalar values arranged in the 3D space, we must map those scalar values to color and opacity using a transfer function, as seen in Figure 1.3. Considering a normalized scalar field, with its values ranging from 0 to 1, the associated color and opacity of a scalar value will be fetched from a RGBA texture, using the scalar as the texture coordinate.

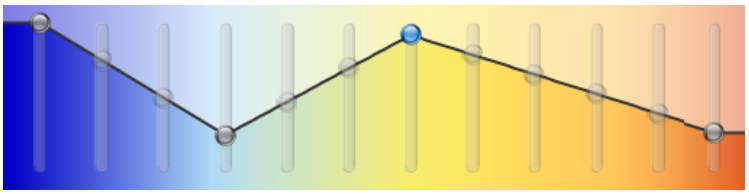
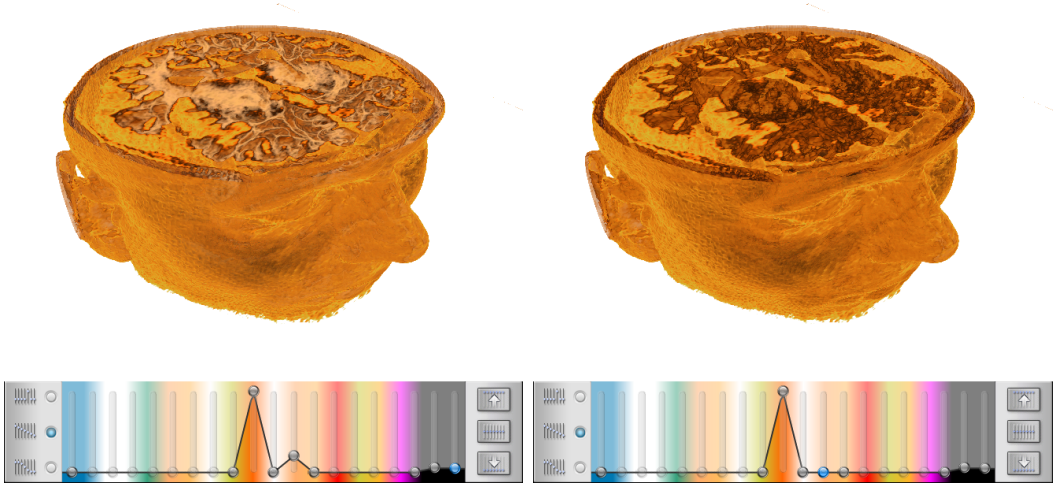


Figure 1.3: An example of a transfer function

Figure 1.4 shows an example of a magnetic resonance imaging dataset modificated through the use of a transfer function.

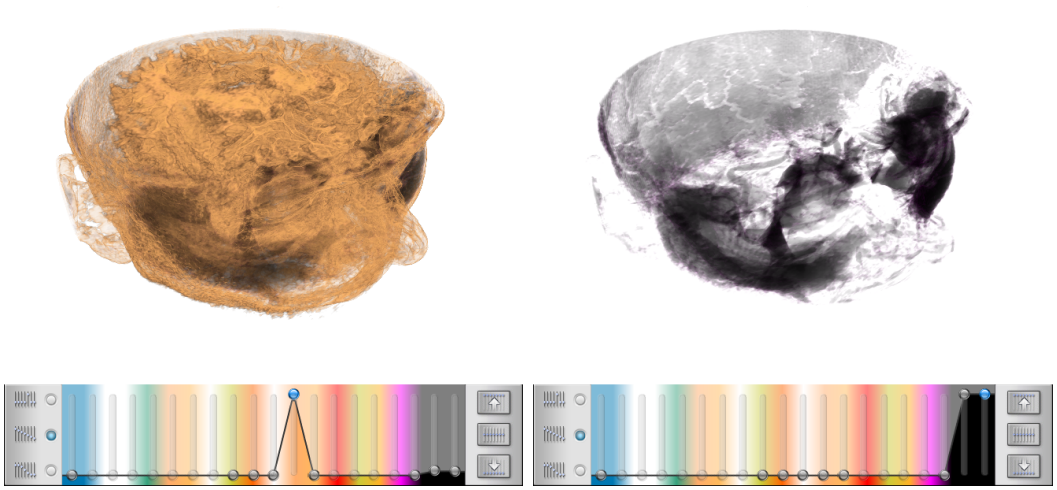
For regular data, as in Figure 1.2(a), it is common to store the volume into a 3D texture and traverse the volume using regular steps, sampling the texture

[1]All volume images in this thesis were taken from our volume renderer.



1.4(a): Transfer function 1

1.4(b): Transfer function 2



1.4(c): Transfer function 3

1.4(d): Transfer function 4

Figure 1.4: Example of four transfer functions on a MRI dataset.

as the ray progresses through the volume. Volume rendering of unstructured data is not so simple; we cannot store the volume into a 3D texture and just sample it, because of the non-uniform cell size. We must then traverse the volume cell-by-cell. As the ray traverses the volume, its color and opacity are calculated taking into consideration the variation of the scalar field inside each cell. As the number of nodes per cell increases, so does the order of the scalar function inside the cell.

For obvious reasons, a linear variation of the scalar field translates into a much simpler interaction between the ray light and the volume. That is why the common approach to render hexahedral meshes is to split each cell into five or six tetrahedra, approximating the trilinear variation of the scalar function by a piecewise linear function. This results in inaccurate image and increases the memory consumption. In this thesis we present a ray-casting

volume rendering algorithm for unstructured hexahedral meshes that considers the trilinear variation of the scalar field inside the cells, and uses a quadrature integration scheme to calculate the interaction between light and volume. We evaluate our solution considering rendering time, image quality, and memory efficiency, and compare it against a hexahedral division solution.

We also present another proposal that approximates the trilinear scalar function by a linear one, but considering certain integration intervals that makes it a better approximation than a hexahedral division approach.

The main contributions of this thesis are:

– A proposal for accurate volume rendering of unstructured hexahedral meshes, considering a trilinear variation of the scalar field.

– A proposal for approximated volume rendering of unstructured hexahedral meshes, considering a linear variation of the scalar field.

This thesis is divided as follow: Chapter 2 presents relevant related work. Chapter 3 presents our proposal for **Accurate Volume Rendering of Unstructured Hexahedral Meshes**, and Chapter 4 our proposal for **Fast Volume Rendering of Unstructured Hexahedral Meshes**. Chapter 5 presents a discussion about the results and Chapter 6 presents our conclusion and future work.