3D Medical Image Segmentation in Virtual Reality

Shea B. Yonker, Oleksandr O. Korshak, Timothy Hedstrom, Alexander Wu, Siddharth Atre, Jürgen P. Schulze

University of California San Diego, La Jolla, CA

Abstract

The possible achievements of accurate and intuitive 3D image segmentation are endless. For our specific research, we aim to give doctors around the world, regardless of their computer knowledge, a virtual reality (VR) 3D image segmentation tool which allows medical professionals to better visualize their patients' data sets, thus attaining the best understanding of their respective conditions.

We implemented an intuitive virtual reality interface that can accurately display MRI and CT scans and quickly and precisely segment 3D images, offering two different segmentation algorithms. Simply put, our application must be able to fit into even the most busy and practiced physicians' workdays while providing them with a new tool, the likes of which they have never seen before.

Introduction

While the idea of displaying medical image stacks in 3D is not a novel concept, we add to it a user-friendly 3D interface for viewing and manipulating them in head-mounted display-based VR, as well as allowing for real-time image segmentation. There is a multitude of software for medical imaging available, but few of them render in 3D stereo with head tracking, and most require heavy amounts of training to use. The ability to manipulate a 3D image to a high degree is there, but the intuitive nature is missing. To get this important tool into the hands of all physicians, especially non-radiologists such as surgeons and primary care physicians, focus must be shifted to the ease of use of such applications. Our research yields a 3D image segmentation tool in virtual reality, which introduces user intuition and 3D input devices while maintaining accuracy, as a solution to the important task of medical image comprehension and feature detection.

To accomplish such a task, our software takes a series of slices (2D medical images taken at differing levels in the body) and stacks them, one on top of the other, in front of the user in the virtual space. By transforming these squares of pixels into a cube of voxels (3D pixels), we can sample the data in 3D space, allowing for a more intuitive representation of the subject of the images. Each voxel is therefore represented by a 3D texture coordinate, a color value, and a mask value. The latter being an on or off value which represents whether the voxel is to be displayed or not.

However, because our data no longer lives in 2D screen space, we need to decide how to determine the color of each pixel on the headset's lenses. To do this we use volumetric ray casting by shooting rays from the eye position out into the data. The voxels intersected by these rays are then sampled and blended to determine the resulting color for the pixel through which the ray was originally shot from.

With the data now mirroring a 3D object and the portion of the body that the scans were initially taken of, we allow the user to reach out in the virtual space and touch the data. They can translate it, rotate it, scale it, and even reach into it to look inside, all by using their hands like they would in the real world. In fact, our application goes beyond simulating what one could do in the real world by allowing the user to reach into the data set as if it is a hologram.

Finally, to more particularly examine one aspect of the data, our program allows for segmentation of this 3D image. For humans, looking at an image and deciphering foreground vs. background is in most cases trivial. Whereas for computers, it can be one of the most difficult and computationally taxing problems. For this reason, we will introduce several segmentation solutions, each of which is tailored to a specific application of medical imaging.

Related Work

As mentioned previously, 3D images, volume rendering, and their segmentation are not new concepts. Some of the most popular and polished 3D image segmentation tools are 3D Slicer, ImageJ, ITK-SNAP, TurtleSeg, and Amira/Avizo. While these have been highly refined to offer hundreds of manipulation options for data sets, they each have many pages of documentation to accompany them, which a user must read through to learn how to successfully utilize their software.

For example, 3D Slicer can be a very useful tool, but only in the hands of someone who knows how to use it. The Slicer Welcome Tutorial [Pujol 2018] alone is twenty-two pages long and teaches you nothing about the application, just on how to use the documentation to answer all the questions you are guaranteed to have in the future. ITK-SNAP on the other hand, claims to feature “seamless 3D navigation”, and “emphasizes interaction and ease of use, with the bulk of the development effort dedicated to the user interface” [Gerig 2018] The effort is quite noticeable, and the application does feature a more forgiving learning curve, but to a seasoned surgeon who has learned to live with traditional 2D scans, there is no room for such a tool in their refined work flow.

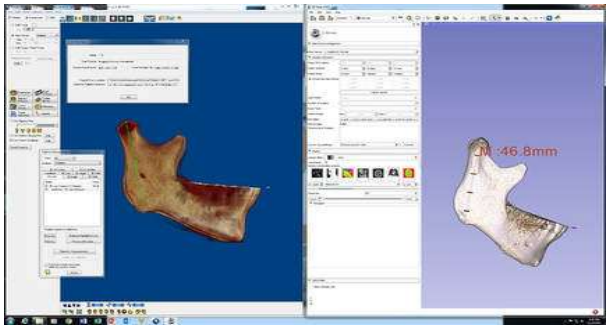

Figure 1: A typical example of the complexity of a 3D Slicer project [Harris2017].

In order to make the cut, the software must be incredibly intuitive and provide something needed and new. While software like ITK-SNAP can make efforts to simplify their user

interface, their user's inability to enter the same space the 3D image occupies creates a separation from the data which requires unintuitive input to accomplish simple tasks.

3D Input Devices

While most of us have become comfortable with a mouse and keyboard, which accomplishe 2D tasks well, such as email checking and Facebook scrolling, when we enter the 3D space sometimes these input methods are insufficient. For instance, when one wishes to cohesively edit a model of a car, the two degrees of freedom of mouse movement along a pad can only cover two dimensions. To indicate a traversal in a third axis would require additional input such as a mouse click or keyboard press, not to mention how one would alter camera or view positioning as well. A simple task such as selecting the moon roof of a 3D car model can require lots of application knowledge.

This is where 3D input devices can make things easier. They can be described as "a form of human-machine interaction where users are able to move and perform interaction in 3D space" [Wikipedia 2018]. While these take many shapes, most allow for users to have access to six degrees of freedom, three for translation and three for rotation. For manipulation of 3D images specifically, the value of these cannot be overstated and the use of one 3D controller allows for easy and intuitive spatial manipulation of the data.

Virtual Reality

With modern advances in graphics cards, access to virtual reality applications has become both affordable to consumers and profitable for businesses who can uniquely incorporate them into their daily workflow. VR offers the ability to view and interact with 3D objects in the same space they occupy, opening the door for every facet of 3D computation to utilize its potential. In essence, VR is not primarily for entertainment anymore.

When these head-mounted displays are coupled with 3D input devices, the user can be transported to the same space as the 3D object they are editing, allowing for incredibly intuitive selection and manipulation of their target objects. It is through this one to one correspondence in editing and display that image segmentation can find new ground for accessibility to all users, and our application strives to use the best this medium has to offer.

Unity 3D

One of the most widely used software programming environments for VR applications today is Unity 3D. Unity's expansive features for image loading, image management, and shader construction have given this project the perfect base for its creation. As opposed to graphics software that is built on top of lower level libraries such as OpenGL, Unity allowed for this entire project to be created in a matter of months, allowing for aspiring computer graphics researchers to study complex components of medical image and virtual reality interface construction.

Representation of Data

By integrating 3D image manipulation with virtual reality, our research removes the barrier of having to mentally stack up many 2D images and allows all users to interact with the data as if it was present in the real world. Each scan of the image will fill layers of a cube in front of the user, thus representing a 3D image as a 3D object in the user's space. These images will only be loaded once, requiring a shader program and the GPU to determine which pixels to draw each frame. This makes the data look accurate quickly, regardless of the viewing perspective.

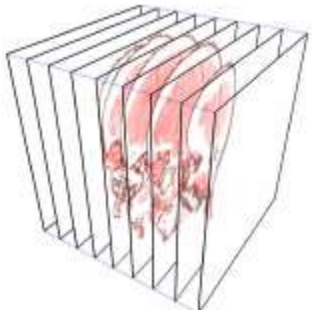

Figure 2: A 3D image (multiple 2D images of equal size and resolution), stacked horizontally (laterally) [Benassarou 2005].

Thinking ahead towards the requirement of segmentation of this data, we associate a mask structure of equal proportions and size. Rather than containing color values, this has a simple toggle, 0 or 1. By multiplying this value with its associated location's color in the 3D image, we can choose to draw or not draw the voxel. By editing this structure when we segment, we only have to set this small value and can avoid needing to alter the entire loaded texture as a whole. This dramatically saves time when segmenting.

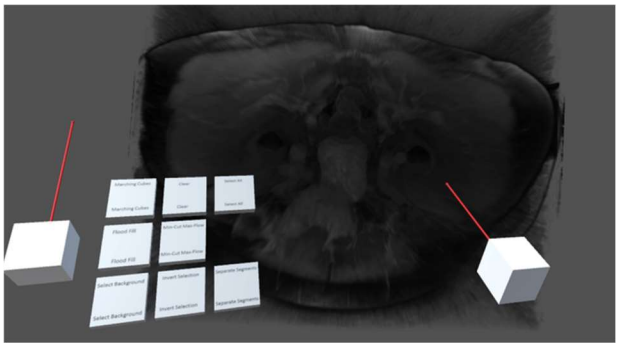

Figure 3: Program running with 128 slices of a colon dataset.

Rendering

When drawing we face the issue of projecting our 3D image representation back onto the 2D screen of the viewer. Specifically in our virtual reality application, we have to determine what color to make each pixel on the lenses of the headset. To do this we use volume ray casting, which can be broken down into the four main steps below.

1.) Ray Casting - For each pixel we send out a ray from the eye position and into the scene by sending it through the pixel. This gives a unique direction for each ray and results in a color for each pixel.

2.) Sampling - With this ray now cast into the scene we determine which voxels were intersected and therefore which will contribute to the resulting pixel color by their proximity to the ray.

3.) Shading - The contributing voxels are then shaded based on their color value from the original scans, multiplied by their mask factor. This ensures that those that are a part of the current segmentation are factored in, and those that are not don't contribute.

4.) Compositing - Finally the left-over voxels' color values are blended to determine the color of the pixel the ray was initially cast through.

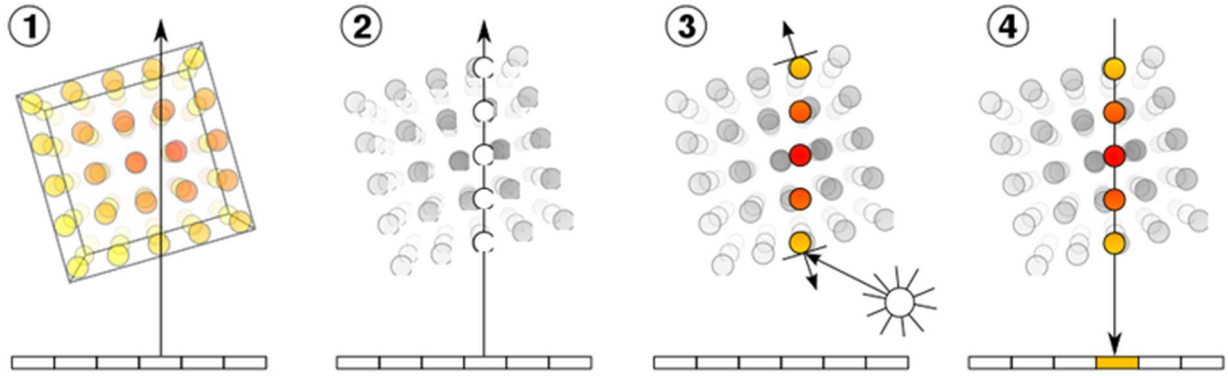

Figure 4: Schematic of our rendering algorithm.

Manipulation

To move the data set, the user can grab it with their 3D input device and its position and rotation will be mapped to that of the user's wrist and hand. By grabbing the data with both hands, they can pull them apart or together to resize the 3D image. To remove complexity, all functions will be accessible from a menu which is always attached to the left hand and within reach of the user.

When reaching into the data, all of the voxels between the user's hand and eye will not be rendered, allowing them to cut into the data with their hand as they go. For example, if the data is of a patient's chest, the doctor can easily examine the heart by extending their arm into the 3D image. They can also position their body and head to view the image from any angle, allowing collaborators to examine data simultaneously from different perspectives.

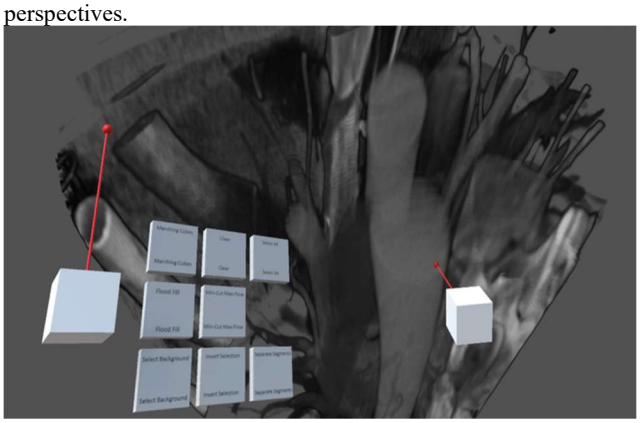

Figure 5: Heart data set with right-hand cutting plane activated.

Segmentation

Segmenting a medical data set is easy when the contrast between desired and surrounding materials is high, such as bone in CT scans. It is much harder when the contrast difference is minimal, such as in MR images without contrast agent. To address the variety in segmentation scenarios, we implemented different segmentation algorithms, each of which have their advantages and disadvantages. In the following sections we describe the algorithms we implemented in our VR image segmentation tool.

Cube Cut

A common manipulation of 3D images is to cut the data so that it removes that which you are not concerned with. To make this as easy to use as possible, when called, this method will start with an interior cube which cuts the data. All that falls inside will be displayed whereas the data outside will not. The user can grab and move this cube, confirming the cut when it contains the desired data. For non-cubic cuts, they can also grab any of the faces of the cube and push or pull them in or out.

Figure 6: Unaltered heart data, initial cube cut, cube cut with adjusted cutoff planes.

Region Grow

With points of origin (seed points) and a threshold specified, this algorithm looks at neighboring voxels and their difference in color value, and if they are within the tolerance, adds them to the segment. On high-contrast medical images, the results of this algorithm are quick and accurate. For tasks such as segmenting large bones or organs, this is the preferred method.

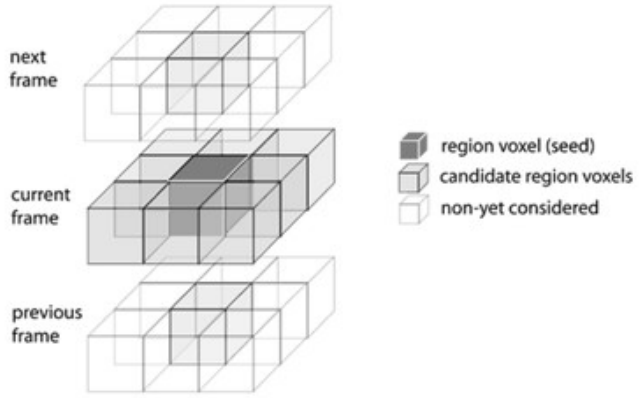

Figure 7: Region Grow algorithm [Kim 2017].

Max-Flow Min-Cut

This approach requires the user to place seed points for both the foreground of what they wish to select (source) and the background (sink). A weighted graph is created where each node corresponds to a voxel from the image data and neighboring voxels are connected by edges with weight inversely proportional to the difference in their color intensities. Two special nodes are added to the graph, a source and a sink, with connections of weight 1 or 0 to the user-specified foreground and background voxel nodes. A minimum graph cut is computed between the source and the sink to segment the image at the frontier with the weakest edge, or sharpest change in color intensity. Because this method of segmentation uses the relative difference between voxel intensities instead of relying on absolute magnitudes it is able to process data with poor contrast and gradual edges.

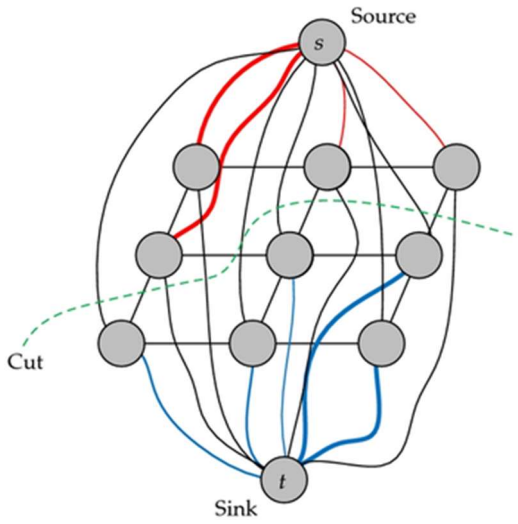

Figure 8: Schematic of the Max-Flow algorithm [Chen 2012].

Conclusion

Virtual Reality provides a one-to-one correspondence between a user's input and the manipulation of their data, allowing for simple access to normally complicated image views. Specifically, the structure of our software provides rapid and accurate 3D image loading, rendering, and segmentation. By shifting the focus of medical imaging away from feature saturation and towards intuitive 3D user interfaces, our application places the powerful tool of 3D image segmentation into the hands of doctors who lack the time to learn the intricacies of today's solutions.

Future Work

Currently we are working on shifting the program logic and segmentation methods to the GPU. While the CPU will still load the data and set up the environment, we are hoping to transition the bulk of the application to shaders. 3D image segmentation and virtual reality are some of the most taxing commands on a computer, and the speed increase gained from this shift in structure will greatly improve the project and open doors for more computationally intensive methods to be added.

However, our largest focus is to bring our application to the point where it is a tool every doctor would be happy to incorporate in their daily process. Segmenting 3D representations of a patient's body in virtual reality can be an incredibly useful tool in the pre-operation and visualization phase. However, it is difficult to picture a surgeon using a headset during the actual operation. This is why we have our sights set on implementing a 2D counterpart which gives surgeons access to saved viewpoints and segmentations from their VR pre-operation stages. Our program offers a unique way to visualize a patient's data set and this would allow physicians to take these views with them into applications where a headset would be a hindrance. As virtual reality offers groundbreaking results for medical imaging while traditional 2D views provide a portable familiar solution, it is the combination of the two that would be an incredibly powerful asset for doctors worldwide.

References

[Benassarou 2005] Benassarou, Aassif & Bittar, Eric & John, Nigel & Lucas, Laurent. (2005). MC Slicing for Volume Rendering Applications. Lecture Notes in Computer Science. 3515. 47-84.

[Boykov 2006] Boykov, Y., & Funka-Lea, G. (2006). Graph cuts and efficient ND image segmentation. International journal of computer vision, 70(2), 109-131.

[Chen 2012] Chen, SY & Tong, Hanyang & Cattani, Carlo. (2012). Markov Models for Image Labeling. Mathematical Problems in Engineering. 2012.

[Gerig 2018] Gerig, G. (2018, April 27). ITK-SNAP Home. Retrieved June 13, 2018, from http://www.itksnap.org/pmwiki/pmwiki.php

[Harris 2017] Peyton Harris (November 28, 2017). Annotation Ruler. Retrieved June 13, 2018, from https://discourse.slicer.org/t/annotation-ruler/1550/3

[Kim 2017] Kim, D., & Chi, T. (2017). 3D Region Growing for CT-Scan Kidney Segmentation. Retrieved April 28, 2018, from http://www.via.cornell.edu/ece547/projects/g3/exper.htm

[Pujol 2018] Pujol, Sonia (2018). Slicer Welcome. Retrieved September 25, 2018, from https://www.slicer.org/wiki/Documentation/4.8/Training#Slicer_Welcome_Tutorial

[Wikipedia2018] 3D Interaction. (n.d.). In Wikipedia. Retrieved March 13, 2018, from https://en.wikipedia.org/wiki/3D_interaction

Author Biography

Dr. Schulze is an Associate Research Scientist at UCSD's Qualcomm Institute, and an Associate Adjunct Professor in the computer science department, where he teaches computer graphics and virtual reality. He holds an M.S. degree from the University of Massachusetts and a Ph.D. from the University of Stuttgart, Germany. After his graduation he spent two years as a post-doctoral researcher in the Computer Science Department at Brown University working on real-time volume rendering in virtual reality.

Shea Yonker, Oleksandr Korshak, Timothy Hedstrom and Alexander Wu and Siddharth Atre were undergraduate students in UCSD's Department of Computer Science when they worked on this project. They have since graduated.