Using the Marching Squares[1] Algorithm to Visualize MRI Brain Textures

Matthew Bennett*, Rick Bischoff‡, Matt Fuqua^

University of Illinois at Urbana-Champaign

ABSTRACT

Traditionally, the Marching Squares algorithm has been used to produce contour lines on topographical maps or what is known as isobars on weather maps. After learning about this technique in our Scientific Visualization class, our team thought it would be interesting if we could apply this algorithm to the medical profession where Magnetic Resonance Images (MRI's) are viewed as part of the diagnostic process. By adding contour lines around objects with higher density and mass to standard two-dimensional MRI brain images, perhaps we could make the presence of tumors and other growths more readily visible and easier to diagnose.

KEYWORDS: Marching Squares, threshold, isolines, Visualization, Brain, Magnetic Resonance Imaging (MRI).

1. INTRODUCTION

During this past semester, our CS519 Scientific Visualization course at the University of Illinois Urbana-Champaign introduced us to the concept of implementing the Marching Squares algorithm[1] in Python, and then applying the algorithm to a two-dimensional image to generate isolines according to a defined threshold value. By looking at the values stored at each of the vertices in the grid, we were able to determine where the isolines should be drawn. The end result, as shown below, is a rudimentary Gaussian image with isolines that represent mock elevations in the diagram. Note that the isolines are closed curves and represent a definable and distinct space on the image.

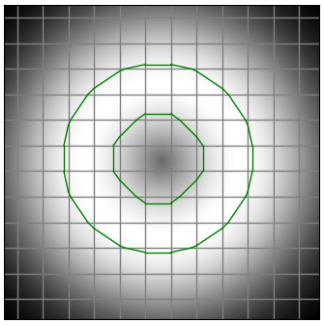


Figure 1: Rudimentary Marching Squares example on generated Gaussian image. Note that the lines were generated at indices where the threshold values were above a predefined level.

If we analyze this rudimentary image, it has a distinct similarity to images that are produced via the MRI process. In the image below, we see an MRI of an adult male head scan. Similar to the image in Figure 1, the black and white image of the human brain is relatively non exciting at first glance. We can definitely make out that it is a dull image of a brain, with enough detail that we can determine the orientation of the head, where the face is located, etc. However, the specific makeup of the brain itself is rather blunted and the matter within the skull is muted, at best.

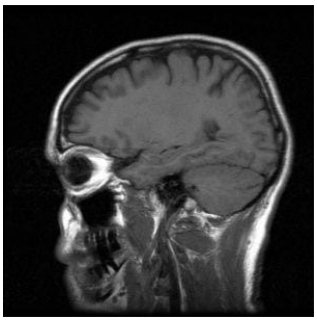


Figure 2: An MRI adult male head scan. Note that while certain details can be made out in the image like the orientation of the head and the location of the eye, the structures of the brain are muted and difficult to see or analyze for any form of diagnosis. (Image copyright © Max Brown https://www.freeimages.com/photo/mri-head-scan-1439617)

For this project, our goal is to take the isoline elements that are shown in Figure 1 and apply the same functionality to images like those shown in Figure 2, with the ultimate goal of providing a visualization of the brain that could potentially lead to easier identification of the elements of the brain and the associated diagnosis of any abnormalities.

2. THE TRANSFORMATION PROCESS

In order to generate the detailed modified images that we were hoping for, we recognized that there would be a series of steps necessary to take our initial simple algorithm and apply if to the complex images produced by the MRI process.

Finding and Understanding the Source Data Files:

The first step required us to locate the source MRI files that we would ultimately run through the Marching Squares algorithm to generate the isolines. After a thorough internet search, we were able to find a couple of sources for MRI data from OpenNEURO at https://openneuro.org and OpenfMRI at https://www.openfmri.org. Within these sites there are numerous public domain files of MRI brain scans that we could use for out testing purposes. The file that we ultimately decided on is located at https://www.openfmri.org/dataset/ds000253/.

As part of the process for analyzing and using these MRI data files in our Python code, we also had to research and understand the metadata associated with these files so that we could accurately apply our algorithm and generate the isolines. The files that we

* email: mwb9@illinois.edu
^ email: mfuqua2@illinois.edu
‡ email: rdb4@illinois.edu

1 https://en.wikipedia.org/wiki/Marching_squares

were able to locate were created using a file format known as NIFTI, which stands for the Neuroimaging Informatics Technology Initiative (https://brainder.org/2012/09/23/the-nifti-file-format/). Files that are stored in this format have a large amount of metadata stored with them to assist with accurately using and analyzing the data files. For our purposes, the most important elements of the metadata were the fields associated with the X and Y spatial dimensions, the number of individual slices of the MRI image that were included, and the value stored at the X/Y intercept grid point that we would ultimately use to compare against a threshold value in our code. Using the Marching Squares algorithm, we could then compare the pre-determined threshold value and use this to determine where the isolines should be drawn.

Creating the Python Code:

Now that the source data files were located and the metadata was processed and understood, the next step in our process was to develop the Python code to read the source file, and then apply the Marching Squares algorithm to each image. The source code that we developed took the initial functions that were developed previously in our CS519 programming assignment as a starting point. From there, modifications were made to include additional Python libraries for plotting and viewing these images, as well as additional modifications to properly implement Marching Squares isoline generator for the MRI files. The completed Python source code and functional description is located at https://github.com/rdbisch/cs519_project/blob/main/Final Version.ipynb.

Vectorized Marching Squares and Performance:

It was feared that the homework-submitted version of Marching Squares was too slow. Further we wanted to use this to process videos and maybe someday serve an interactive use. We spent some time figuring out how to make the code faster and more conducive to these activities.

The first optimization was to rewrite portions of the Marching Squares implementation to operate all at once. That is, rather than going pixel by pixel in the image to determine case, we could use Numpy-Arrays to do it in one fell swoop.

Function Name	Homework Version	Vectorized Version for Project	Speedup (x)
getContourCase	0.106	0.001	104.092
getCellSegments	0.110	0.035	3.148

The 2nd optimization was to rewrite the graphics rendering of the isolines themselves. The original code used matplotlib and was quite slow. One example took 163 seconds to process the video, for an average of 0.850 seconds per frame. We rewrote the code using OpenCV2 and our own rasterization routines. These are not as general purpose as Matplotlib's but most of that functionality is not needed for this. With this optimization, the same video took 40 seconds for an average of 0.206 seconds per frame.

The overall speedup was approximately 4x. Further optimizations could be achieved that are relatively straightforward. We could trace out isolines and have them represented internally as polygons instead of cell-segments. It is also possible to realize some speed-ups by doing rasterization at the same time as isoline generation. This would only make sense to do if you are purely using the isolines as a visual diagnosis tool rather than as an input feature for automatic classification.

Initial Findings:

After successfully creating our Marching Squares code, the first modified MRI images were ready to be evaluated. Note that as part of our algorithm, we applied a color map to the images to enhance the ability to see our white isolines. What we initially found was that the value that we were using for our isoline threshold was a critical factor in seeing any real value in applying the algorithm. If the threshold value was too low, there was very little change in the processed file, but if the threshold value was too high there would be significant static with too many isolines, and the image became unusable. To get a better handle on the quality of the images, we produced several iterations of the output files using various levels of the isoline threshold value. The images that we created looked like this:

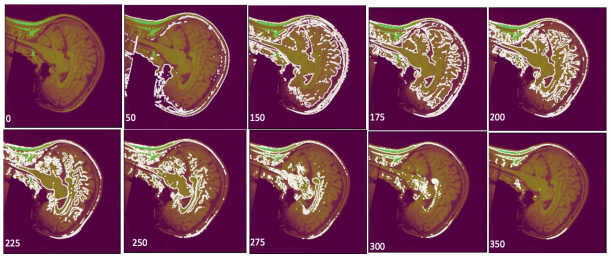


Figure 3: MRI Brain Images processed with Marching Squares algorithm and varying threshold levels (shown in white).

As is depicted in Figure 3, the various threshold levels make a significant impact on the usefulness of applying the Marching Squares algorithm. However, it should also be noted that our hypothesis of applying the algorithm to MRI data did significantly highlight the structures within the brain and made the results easier to see and potentially diagnose. A larger example of these images makes this clearer:

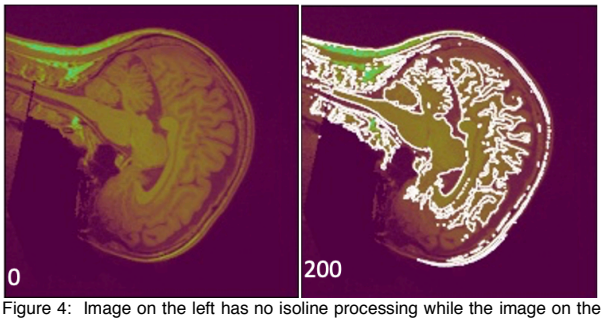
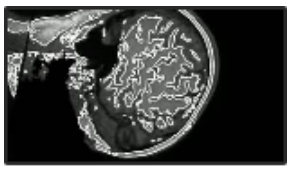


Figure 4: Image on the left has no isoline processing while the image on the right uses a threshold of 200 for isoline generation. Notice that the elements of the brain in the right image are more clearly marked with the white isoline strucures.

Additionally, by taking the individual slices of the MRI that had been processed by our algorithm and compiling them into an animation, you can clearly see even more definition of the brain's elements (click on the link to view the animation):



https://youtu.be/68rwzIqh-fE

Figure 5: **click** on the link to view the animation of MRI slices that have been processed through the Marching Squares algorithm.

3. Specific Potential Uses:

When we reviewed the results of our Marching Squares implementation on MRI data, one of the potential uses we surmised in addition to simply making the scans more readable was the possible early detection of abnormalities like tumors or large growths. To test that theory, we took a single MRI image that contained a large brain tumor and applied our Marching Squares algorithm with the following results:

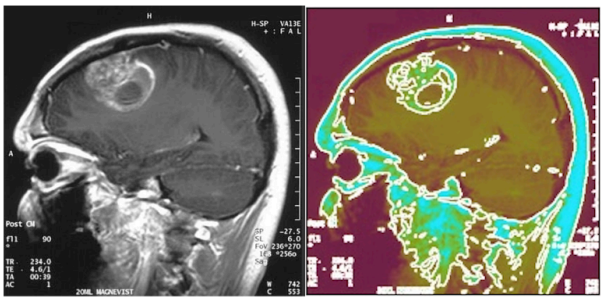


Figure 6: MRI with brain tumor on left and results of our algorithm on the right. Note that the large mass is clearly detected and surrounded by isolines as a result of our algorithm. The image on the right is from https://www.nih.gov/news-events/nih-research-matters/subtypes-deadly-brain-cancer-identified, and is Copyright Christaras A, courtesy of Wikimedia.org.

The output from our algorithm completely encases the large tumor in the brain and does provide some greater visibility to the mass for the viewer. As with the other images, we modified the threshold value throughout a range until the best fit value was determined to show the tumor the most clearly.

4. Conclusion:

While we recognize that the application of our Marching Squares algorithm is not worthy of a Nobel Prize in Medicine, we were very pleased to see that applying the isoline generating code to the MRI files did result in easier identification of the structures within the brain. It is very satisfying to be able to take the lessons learned in the classroom and actually be able to apply it to real-world applications. As we concluded the project we know that doctors are highly educated medical practitioners who are trained to specifically read and understand MRI scans, but we are happy to have gained a better understanding of how the MRI files are constructed and how image files such as these are able to be processed with additional algorithms to further and enhance their clarity.

5. References and/Or Links Mentioned in the Document:

[1] Marching Squares Algorithm Discussion: https://en.wikipedia.org/wiki/Marching_squares

[2] Figure 2 Image Link: https://www.freeimages.com/photo/mri-head-scan-1439617

[3] Link to OpenNEURO: https://openneuro.org

[4] Link to OpenfMRI: https://www.openfmri.org

[5] Link to MRI Source File: https://www.openfmri.org/dataset/ds000253/

[6] Link to Neuroimaging Informatics Technology Initiative website https://brainder.org/2012/09/23/the-nifti-file-format/

[7] Link to Python code for this project: https://github.com/rdbisch/cs519_project/blob/main/Final Version.ipynb.

[8] Link to Youtube animation of Brain Scan: https://youtu.be/68rwzIqh-fE

[9] Link to image with Brain Tumor: https://www.nih.gov/news-events/nih-research-matters/subtypes-deadly-brain-cancer-identified