ATI 3D Aneurysm

Hardware Review

Ahmed Almutairi

Scott Barberii

Jackson Tomaszewski

Austin Vest

ME 486C, Spring 2021

Project Sponsor: ATi

Faculty Advisor: Dr. Tim Becker, Chris Settanni

Instrutor: Dr. David Trevas

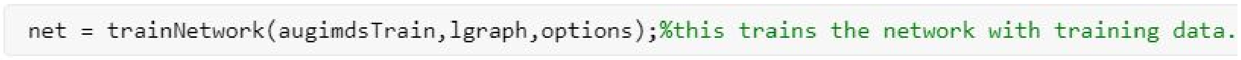
To: Dr. Oman, Dr. Becker, and Chris Settanni
From: ATI 3D Aneurysm Capstone Team
Subject: Hardware Review 1- Current State or Project

Austin Vest- Neural Network

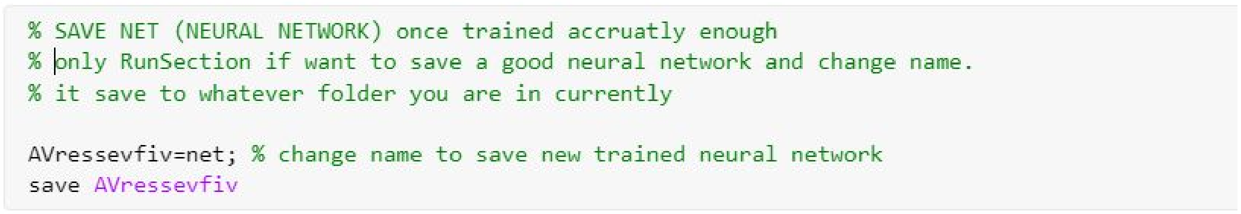
The current state of the neural network:

- it can be trained with any training data,
- the trained neural network can be saved,
- it can be tested on any test images and an accuracy calculated if labeled sub folder,
- An independent code can run separately if a trained neural network is saved.

The neural network training code is long and takes time but once trained to an acceptable satisfactory level, it can be saved as a matlab file to any location seen in Figure A1. Then a separate classification code can load that trained neural network and (classify or label) new test images that have aneurysms in some of them as seen in Figure A2. This step is critical to making a separate independent program. The trained neural network can be downloaded in the main program folder the team is ultimately working to compile.

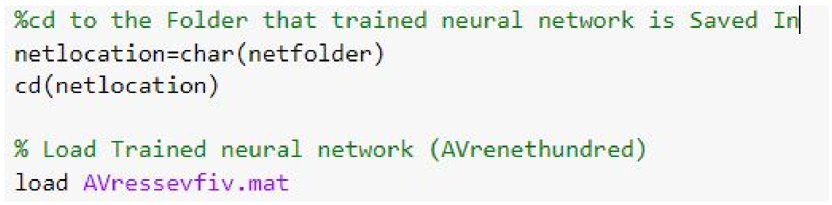


```
net = trainNetwork(augimdsTrain,lgraph,options);%this trains the network with training data.
```



```
% SAVE NET (NEURAL NETWORK) once trained accruatly enough
% only RunSection if want to save a good neural network and change name.
% it save to whatever folder you are in currently

AVressevfiv=net; % change name to save new trained neural network
save AVressevfiv
```

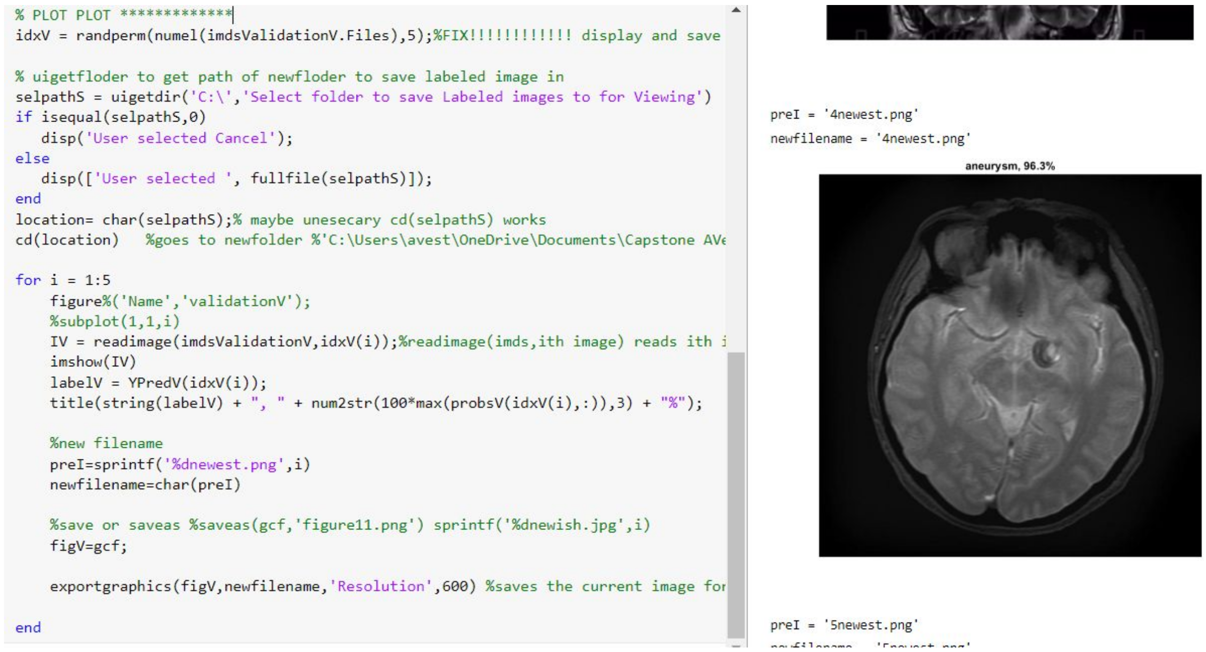
Figure A1: Code that saves Accurate neural network when needed



```
%cd to the Folder that trained neural network is Saved In
netlocation=char(netfolder)
cd(netlocation)

% Load Trained neural network (AVrenethundred)
load AVressevfiv.mat
```

Figure A2: Loading Saved trained neural network in an independent code

The next section of the code is the classify / label images, display those labeled images, and save them to a new folder. The code currently has a “uigetdir” function where the user must select the folder that the new scans are in and a new folder where the user wants to save the labeled images. This allows the user to save the labeled images for future reference and for the team’s GUI to reference. The code for this is also long but involves a “for loop” that plots and saves the labeled images with the labels and confidence percentages gained from the classification completed previously by the neural network. This code and the output image can be seen in Figure A3. The images save in Newfolder can be seen in Figure A4.



```
% PLOT PLOT *************
idxV = randperm(numel(imdsValidationV.Files),5);%FIX!!!!!!!!!!!! display and save

% uigetfloder to get path of newfloder to save labeled image in
selpathS = uigetdir('C:\','Select folder to save Labeled images to for Viewing')
if isequal(selpathS,0)
   disp('User selected Cancel');
else
   disp(['User selected ', fullfile(selpathS)]);
end
location= char(selpathS);% maybe unesecary cd(selpathS) works
cd(location)   %goes to newfolder %'C:\Users\avest\OneDrive\Documents\Capstone AVe

for i = 1:5
    figure%('Name','validationV');
    %subplot(1,1,i)
    IV = readimage(imdsValidationV,idxV(i));%readimage(imds,ith image) reads ith i
    imshow(IV)
    labelV = YPredV(idxV(i));
    title(string(labelV) + ", " + num2str(100*max(probsV(idxV(i),:)),3) + "%");

    %new filename
    preI=sprintf('%dnewest.png',i)
    newfilename=char(preI)

    %save or saveas %saveas(gcf,'figure11.png') sprintf('%dnewish.jpg',i)
    figV=gcf;

    exportgraphics(figV,newfilename,'Resolution',600) %saves the current image for

end
```

```
preI = '4newest.png'
newfilename = '4newest.png'
```

```
preI = '5newest.png'
```

Figure A3: “for loop” that plots and save the image with its label and confidence

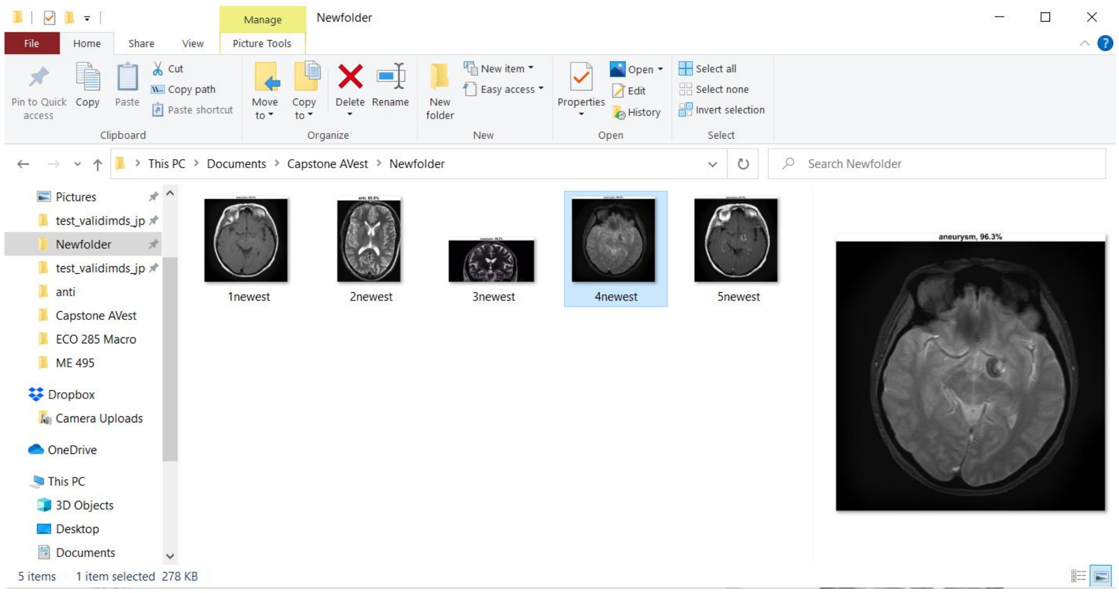
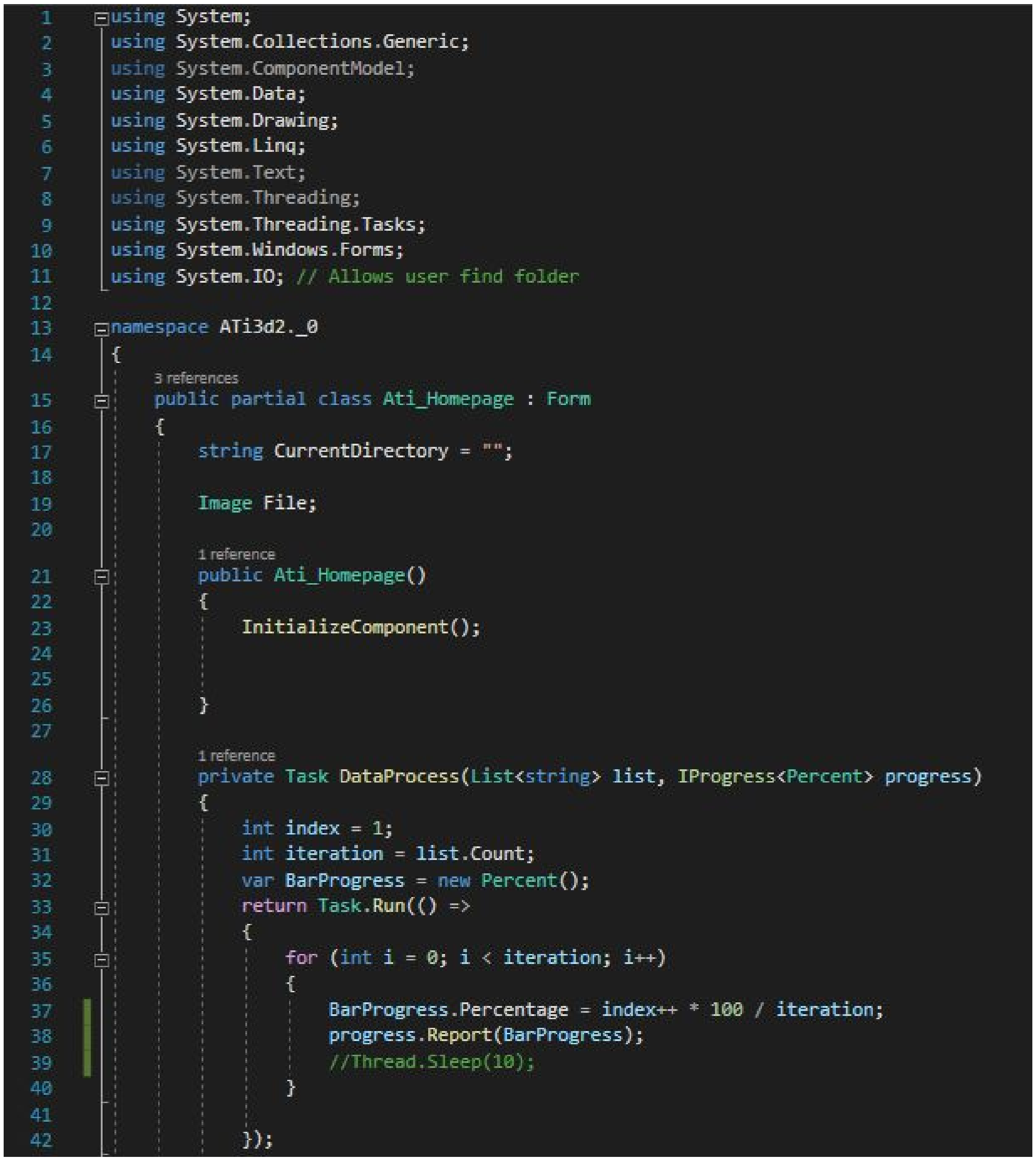


Figure A4: Newfloder that labeled images are saved in.

This team will continue to test different methods and data sets to train a robust neural network that has the best possible accuracy. The training data and be added to, to include MRI, MRA, and CT scan formats (these are the main scanning methods used in the medical industry). The neural network may be able to be trained multiple times although this may not increase accuracy but just change the connections in the neural network. The testing data can also be increased or decreased with one type of scan or multiple types. This part of the project is getting to an acceptable point where this team member's time can be spent more on the GUI and Volume viewer and calculator as needed. This is after the .dicom to .png converter is debugged.

Jackson GUI

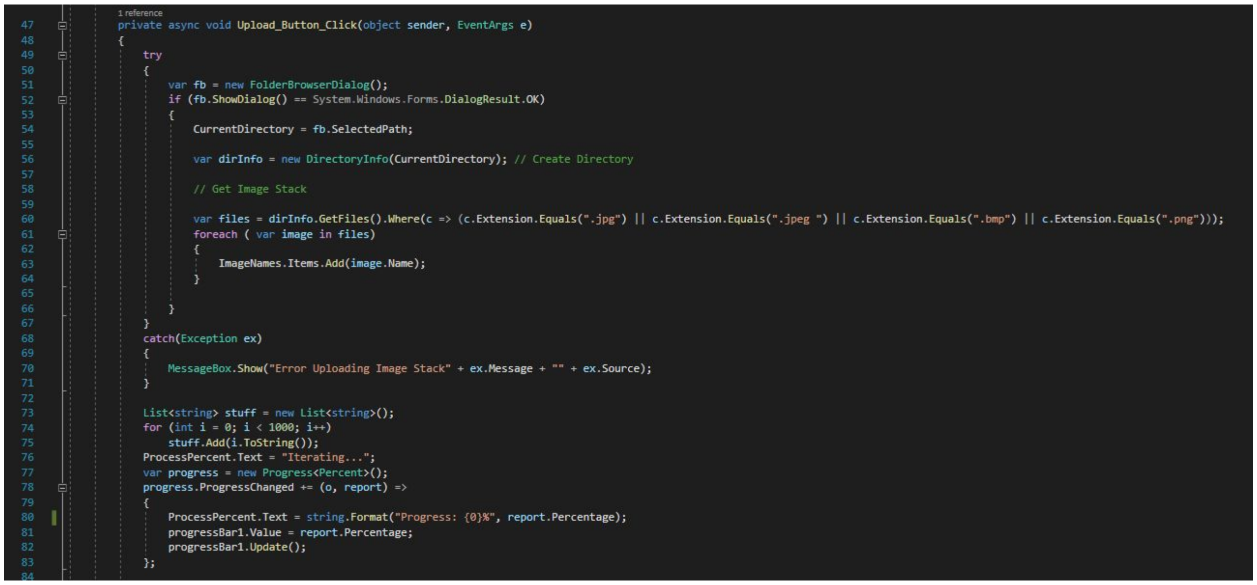
Visual studio was used to create the graphical user interface for our capstone project. It was used because building a GUI in matlab has limited built in functions and the look is rather rudimentary. Below is the opening C# code for the program.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO; // Allows user find folder

namespace ATi3d2._0
{
    3 references
    public partial class Ati_Homepage : Form
    {
        string CurrentDirectory = "";

        Image File;

        1 reference
        public Ati_Homepage()
        {
            InitializeComponent();


        }

        1 reference
        private Task DataProcess(List<string> list, IProgress<Percent> progress)
        {
            int index = 1;
            int iteration = list.Count;
            var BarProgress = new Percent();
            return Task.Run(() =>
            {
                for (int i = 0; i < iteration; i++)
                {
                    BarProgress.Percentage = index++ * 100 / iteration;
                    progress.Report(BarProgress);
                    //Thread.Sleep(10);
                }

            });
```

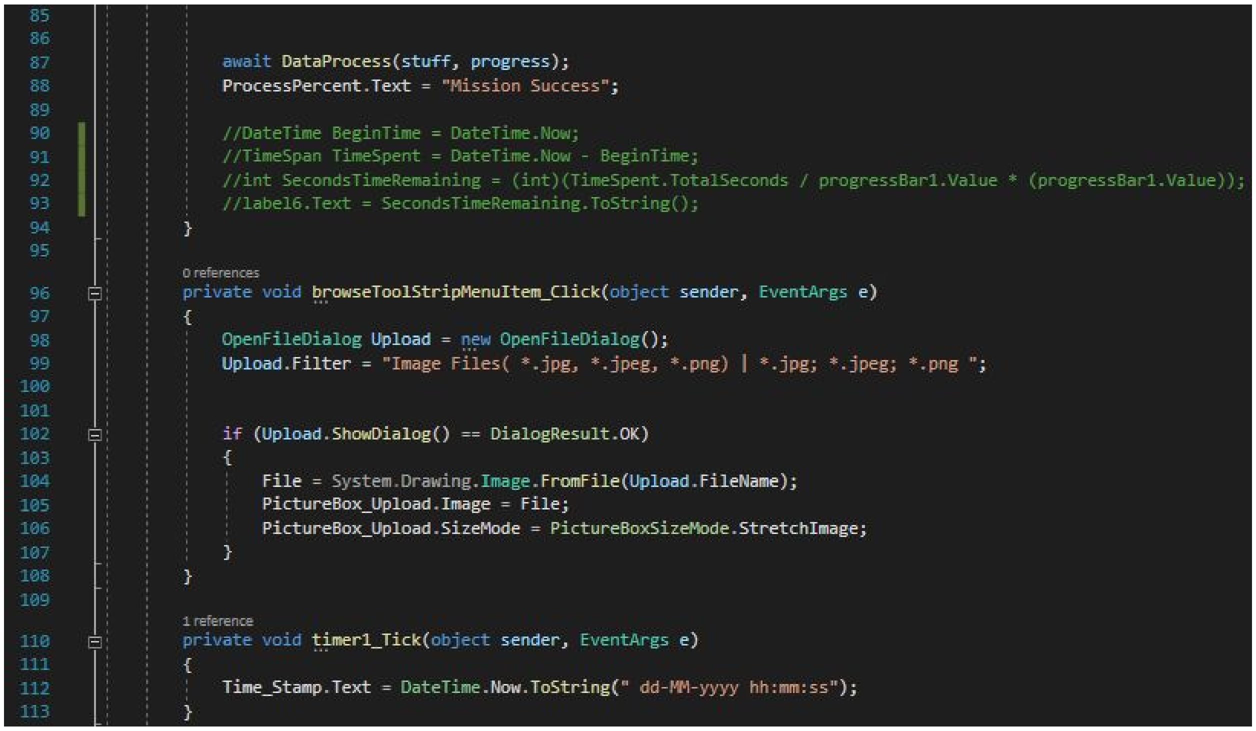
Figure 5: Process Bar

The code above in Figure 5 is initializing functions and is creating a progress bar. The code counts through a 100 different iterations until it reaches 100. It will reach this value when the upload is complete. The counting is necessary to ensure the final value is equal to 1. Figures 6 and 7 below shows which is associated with the upload button. The button can be found in Figure 8.



```
1 reference
private async void Upload_Button_Click(object sender, EventArgs e)
{
    try
    {
        var fb = new FolderBrowserDialog();
        if (fb.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        {
            CurrentDirectory = fb.SelectedPath;

            var dirInfo = new DirectoryInfo(CurrentDirectory); // Create Directory

            // Get Image Stack

            var files = dirInfo.GetFiles().Where(c => (c.Extension.Equals(".jpg") || c.Extension.Equals(".jpeg ") || c.Extension.Equals(".bmp") || c.Extension.Equals(".png")));
            foreach ( var image in files)
            {
                ImageNames.Items.Add(image.Name);
            }

        }
    }
    catch(Exception ex)
    {
        MessageBox.Show("Error Uploading Image Stack" + ex.Message + "" + ex.Source);
    }

    List<string> stuff = new List<string>();
    for (int i = 0; i < 1000; i++)
        stuff.Add(i.ToString());
    ProcessPercent.Text = "Iterating...";
    var progress = new Progress<Percent>();
    progress.ProgressChanged += (o, report) =>
    {
        ProcessPercent.Text = string.Format("Progress: {0}%", report.Percentage);
        progressBar1.Value = report.Percentage;
        progressBar1.Update();
    };
```

Figure 6: Submit Button Code



```


    await DataProcess(stuff, progress);
    ProcessPercent.Text = "Mission Success";

    //DateTime BeginTime = DateTime.Now;
    //TimeSpan TimeSpent = DateTime.Now - BeginTime;
    //int SecondsTimeRemaining = (int)(TimeSpent.TotalSeconds / progressBar1.Value * (progressBar1.Value));
    //label6.Text = SecondsTimeRemaining.ToString();
}

0 references
private void browseToolStripMenuItem_Click(object sender, EventArgs e)
{
    OpenFileDialog Upload = new OpenFileDialog();
    Upload.Filter = "Image Files( *.jpg, *.jpeg, *.png) | *.jpg; *.jpeg; *.png ";


    if (Upload.ShowDialog() == DialogResult.OK)
    {
        File = System.Drawing.Image.FromFile(Upload.FileName);
        PictureBox_Upload.Image = File;
        PictureBox_Upload.SizeMode = PictureBoxSizeMode.StretchImage;
    }
}

1 reference
private void timer1_Tick(object sender, EventArgs e)
{
    Time_Stamp.Text = DateTime.Now.ToString(" dd-MM-yyyy hh:mm:ss");
}
```

Figure 7: Submit Button Code Continued

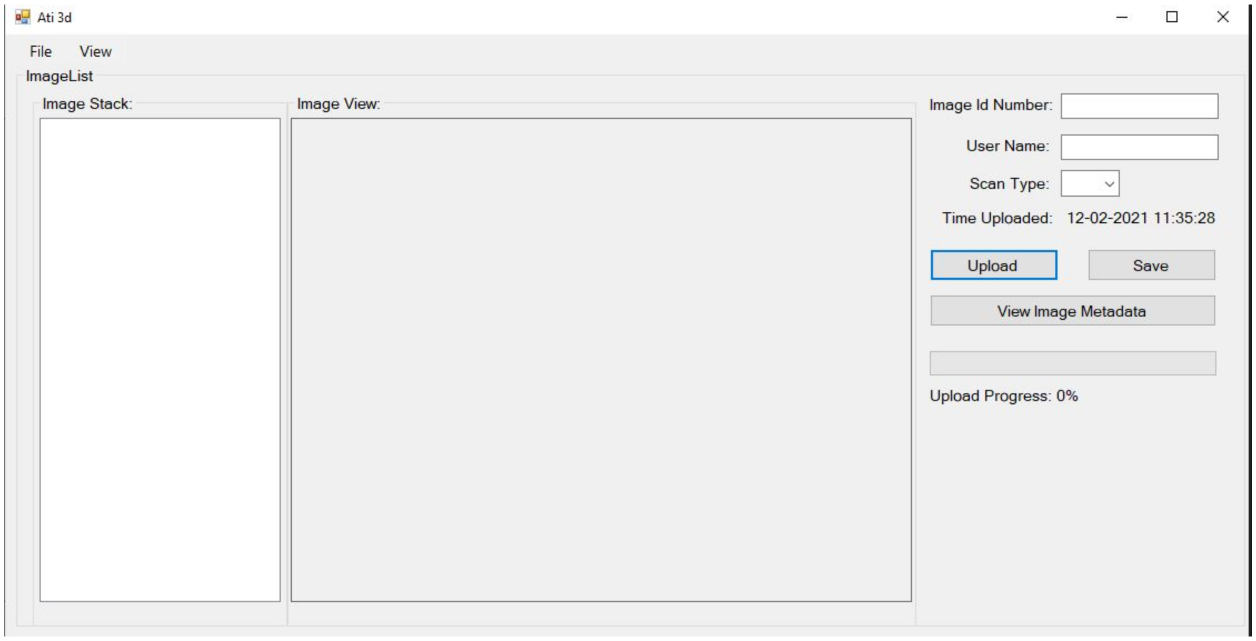
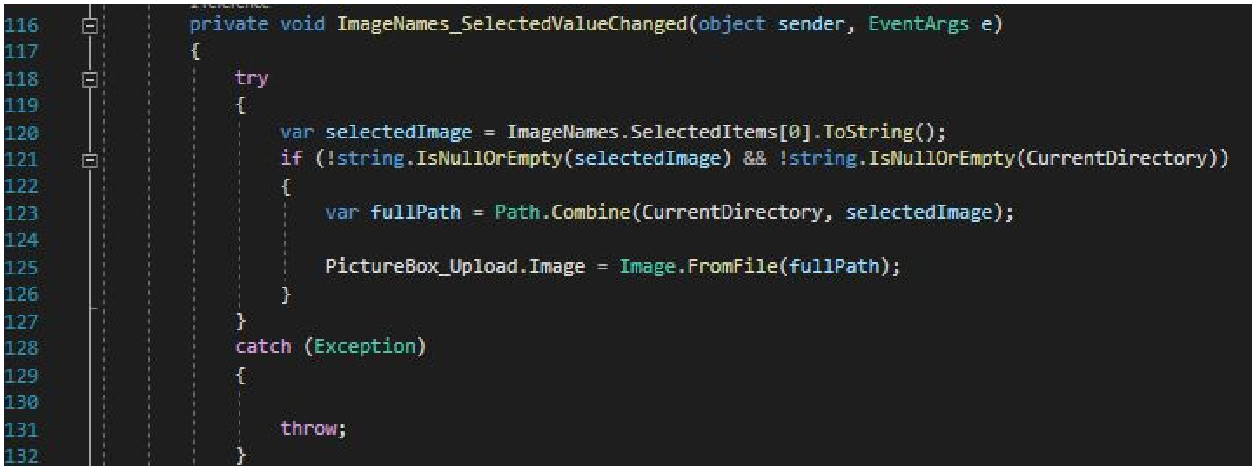


Image Id Number:
User Name:
Scan Type:
Time Uploaded: 12-02-2021 11:35:28
Upload Progress: 0%

Figure 8: GUI Upload Button

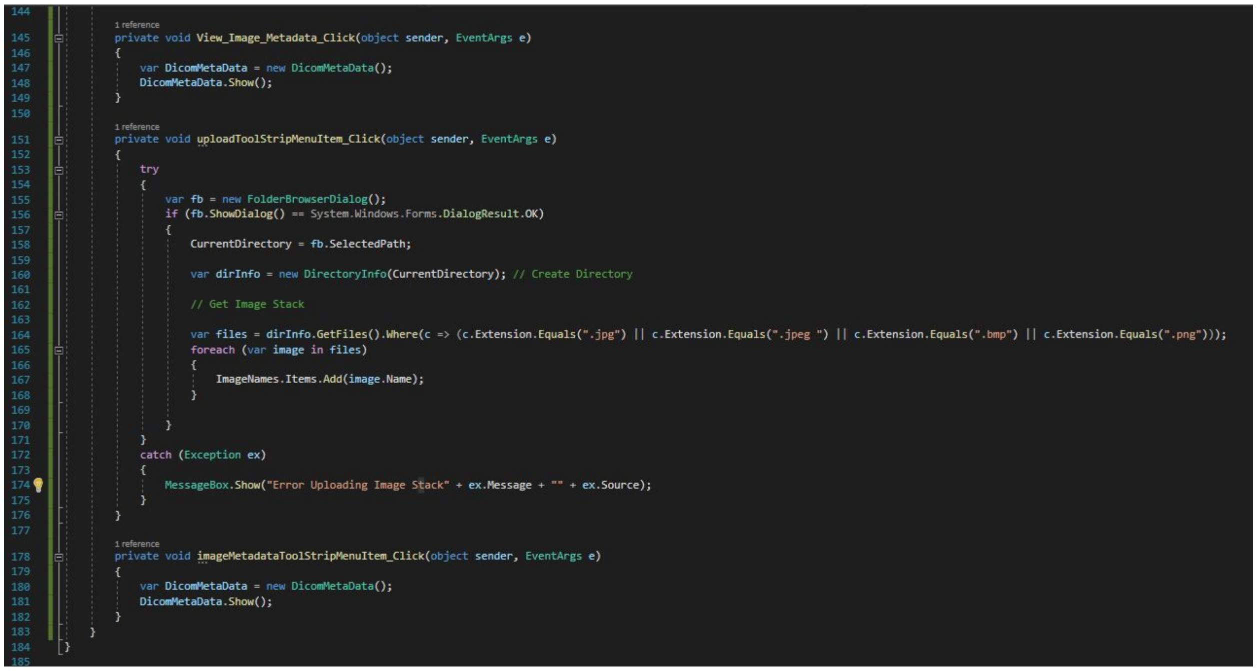
Figures 6 and 7 show the code associated with the upload button. This code will be run upon button click by the user. The first section of the code allows the user to choose a file from the computer files. This section is also filtered to show files with the ending of .jpeg, .jpg, .png, and .png. The section starts the progress bar and competes a few more iteration steps. Figure 7 also has code to create a timestamp which then will eventually be recorded every time the program is run. In addition to that the uploaded files are displayed in the list box on the right hand side of the GUI.



```
private void ImageNames_SelectedValueChanged(object sender, EventArgs e)
{
    try
    {
        var selectedImage = ImageNames.SelectedItems[0].ToString();
        if (!string.IsNullOrEmpty(selectedImage) && !string.IsNullOrEmpty(CurrentDirectory))
        {
            var fullPath = Path.Combine(CurrentDirectory, selectedImage);

            PictureBox_Upload.Image = Image.FromFile(fullPath);
        }
    }
    catch (Exception)
    {

        throw;
    }
```

Figure 9: Files to Picture Box

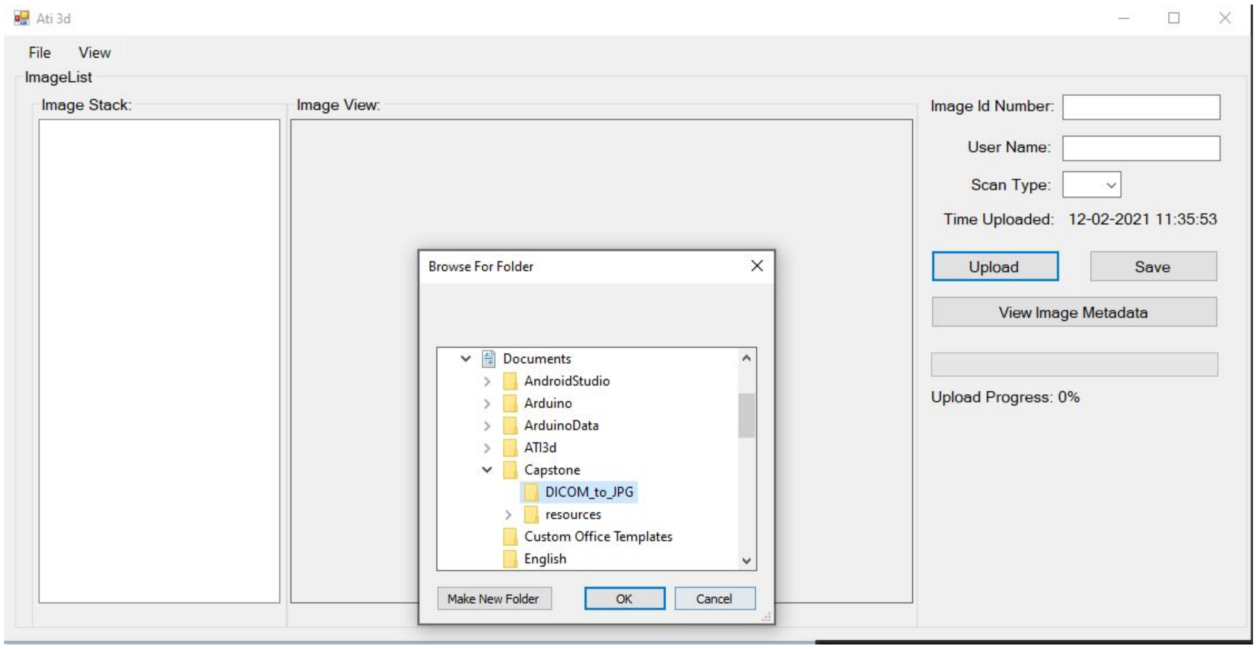
Figure 9 above is the code that displays the selected images to the center picture box. When a new image is selected the section will then be overridden and the new image will be displayed. Below in Figure 10 is the code that allows the user to use the menu strip and metadata button.



```
        1 reference
        private void View_Image_Metadata_Click(object sender, EventArgs e)
        {
            var DicomMetaData = new DicomMetaData();
            DicomMetaData.Show();
        }

        1 reference
        private void uploadToolStripMenuItem_Click(object sender, EventArgs e)
        {
            try
            {
                var fb = new FolderBrowserDialog();
                if (fb.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    CurrentDirectory = fb.SelectedPath;

                    var dirInfo = new DirectoryInfo(CurrentDirectory); // Create Directory

                    // Get Image Stack

                    var files = dirInfo.GetFiles().Where(c => (c.Extension.Equals(".jpg") || c.Extension.Equals(".jpeg ") || c.Extension.Equals(".bmp") || c.Extension.Equals(".png")));
                    foreach (var image in files)
                    {
                        ImageNames.Items.Add(image.Name);
                    }

                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error Uploading Image Stack" + ex.Message + "" + ex.Source);
            }
        }

        1 reference
        private void imageMetadataToolStripMenuItem_Click(object sender, EventArgs e)
        {
            var DicomMetaData = new DicomMetaData();
            DicomMetaData.Show();
        }
    }
}
```

Figure 10: Menu Bar

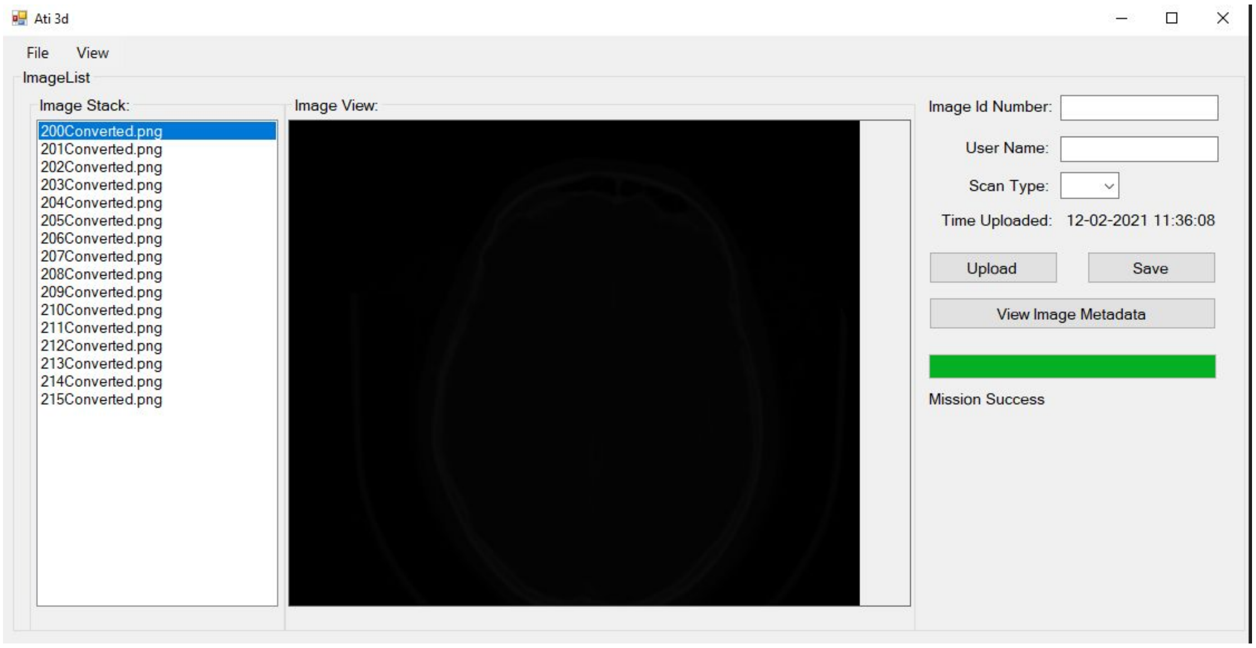
Figure 10 is the code which allows the user to interact with the menu strip. It is basically the same exact code used in Figure 6. Except it is for a menu bar and not the submit button. The last few lines of code are the for the metadata button. It creates a new window that will eventually display a the metadata for the selected image.



Time Uploaded: 12-02-2021 11:35:53

Upload Progress: 0%

Figure 11: Browse Files



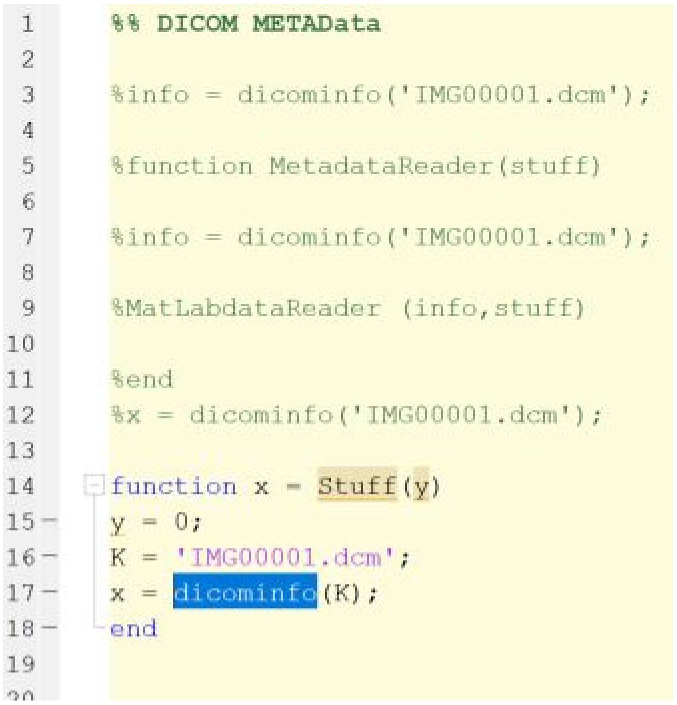
200Converted.png
201Converted.png
202Converted.png
203Converted.png
204Converted.png
205Converted.png
206Converted.png
207Converted.png
208Converted.png
209Converted.png
210Converted.png
211Converted.png
212Converted.png
213Converted.png
214Converted.png
215Converted.png

Time Uploaded: 12-02-2021 11:36:08

Mission Success

Figure 12: File Names

Figure 11 and 12 above are the file browser and the complete action after selecting the folder. The names are displayed on the left side and the images are displayed in the center. Below in Figure 13 is the MATLAB code that will be used to display the DICOM images metadata.



```
%% DICOM METAData

%info = dicominfo('IMG00001.dcm');

%function MetadataReader(stuff)

%info = dicominfo('IMG00001.dcm');

%MatLabdataReader (info,stuff)

%end
%x = dicominfo('IMG00001.dcm');

function x = Stuff(y)
y = 0;
K = 'IMG00001.dcm';
x = dicominfo(K);
end
```

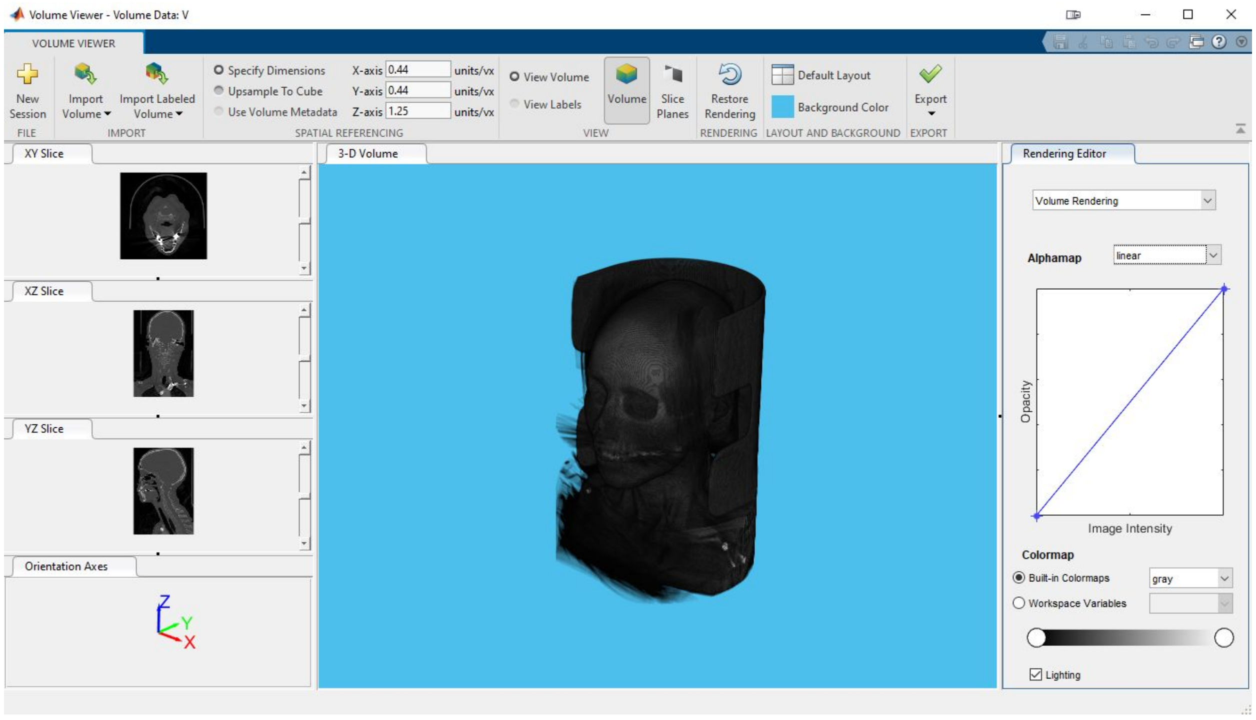
Figure 13: MATLAB Code

Above is a function that can be used in Visual Studio. Line 17 is the built in matlab command which extracts the metadata from the DICOM file. This must be in the form of a function because there must be only one entry point for the function. After this the file can be converted using the matlab library compiler. This converts the file into .NET format.

Ugrad video or poster submission to a panel of judges
Wide audience
U grad presentation freshman sophomore
Poster or video concise and conveying time spent and important code parts
I need to do 495 HW. I think my section is good. Let me know before we submit.

Scott and Ahmed- 3D volume

Scott-



X-axis 0.44 units/vx
Y-axis 0.44 units/vx
Z-axis 1.25 units/vx
Volume Rendering
Alphamap linear
Built-in Colormaps gray

Figure 14: Volume viewer
This is the screen shot from the volume view built into matlab to see a dicom file.

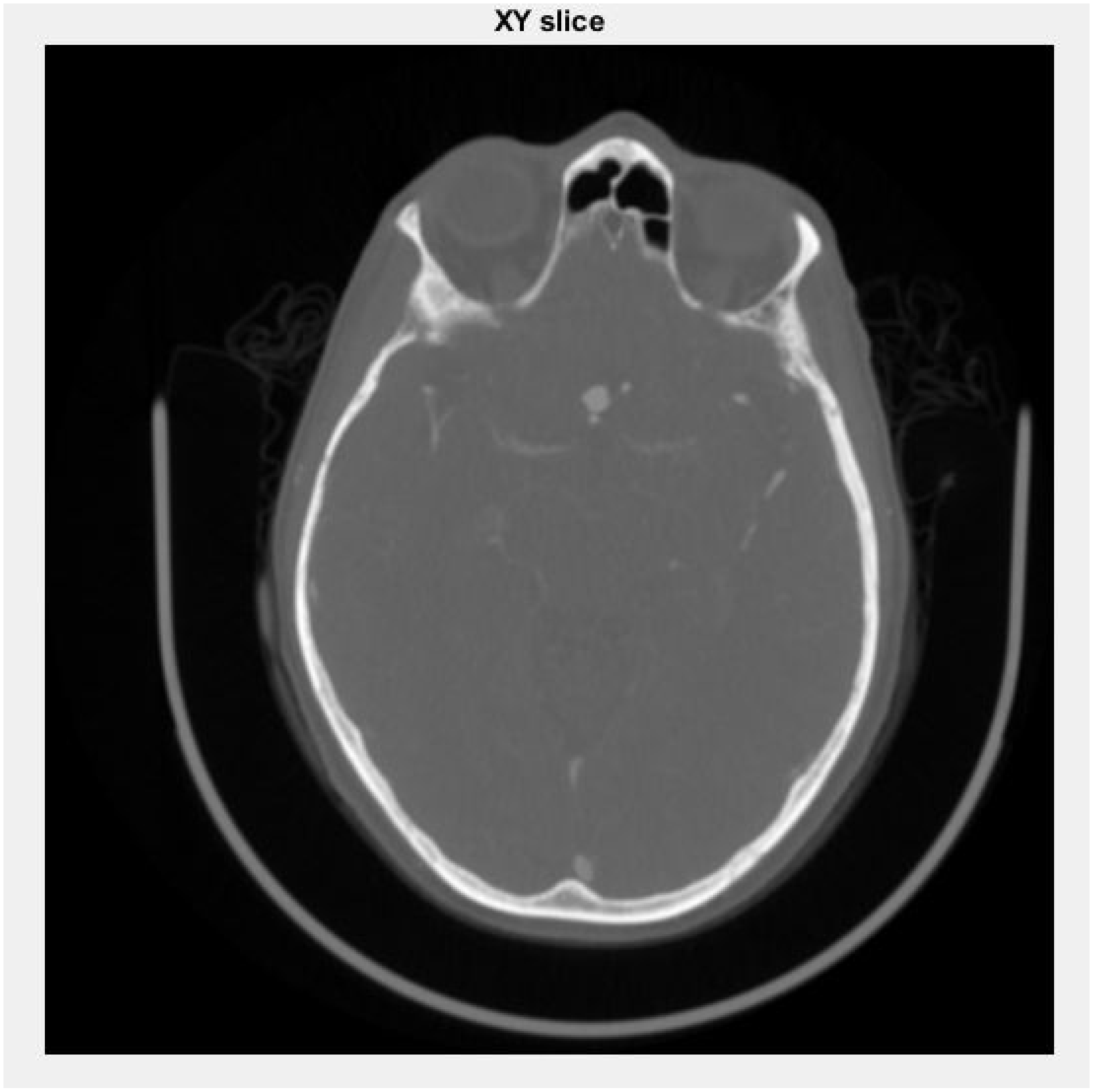


Figure 15: XY slice

Here is an example image that would be chosen if it had an aneurysm in it, this slice is in the XY plane while the other slice is from the XZ plane.

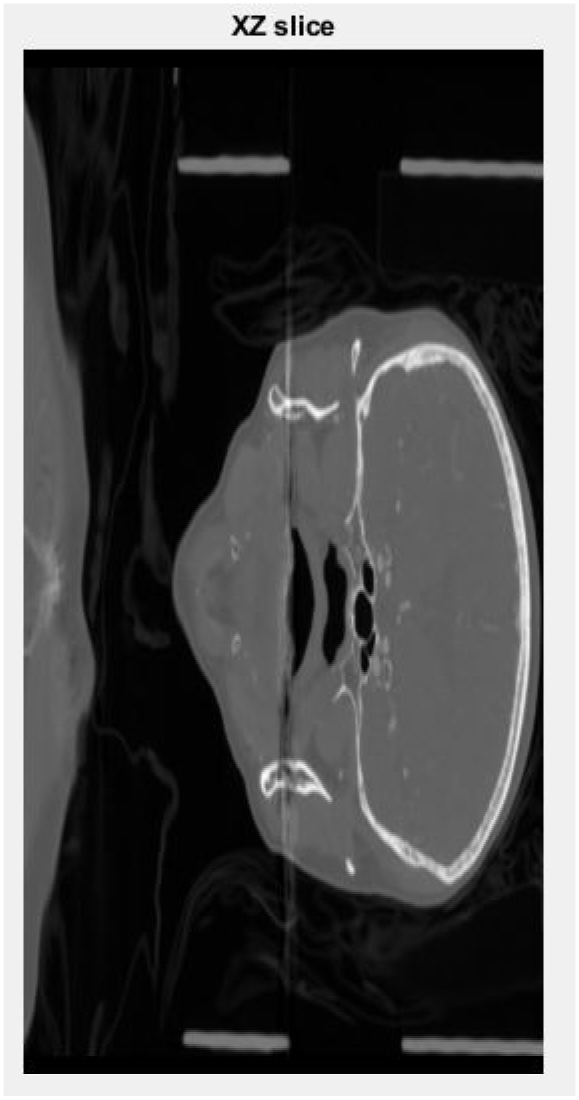


Figure 16: XZ slice

Next we have the active contours menu where the user would highlight where they believe the aneurysm to be. We had one issue with location of aneurysm in XZ slice , we need to improve the location in XZ slice in order to get accuracy results.

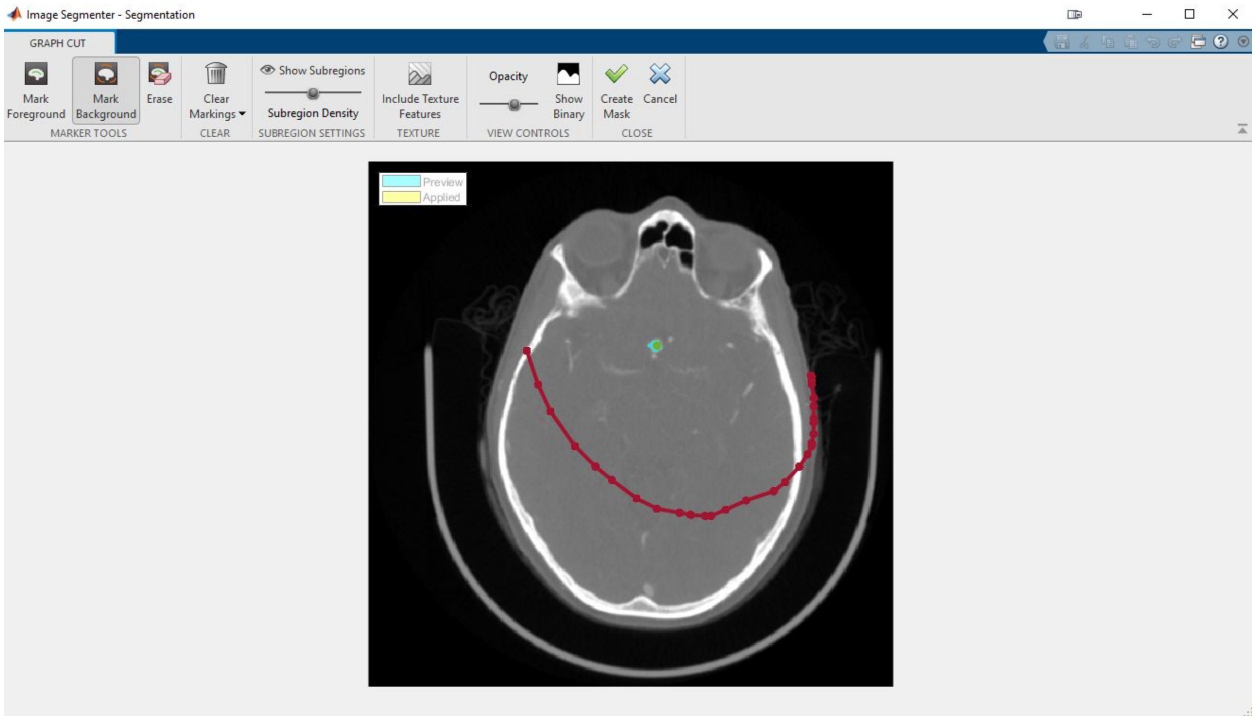


Figure 17: Image Segmentation

Finally after selecting the aneurysm in both planes, using the function active contours, Matlab can create a 3D rendering of just the selection, in this case the aneurysm.

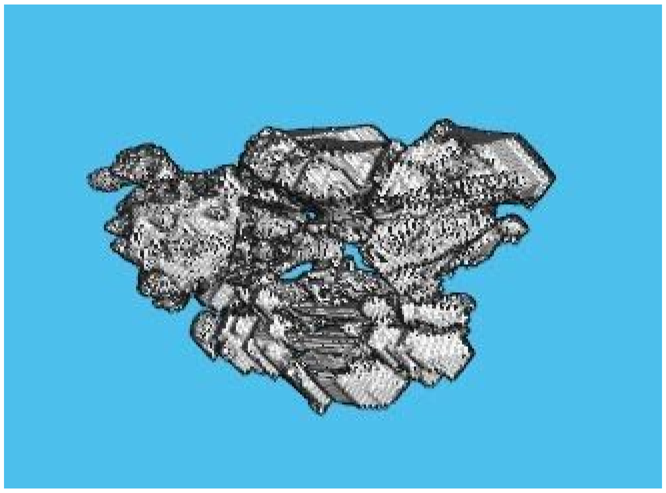
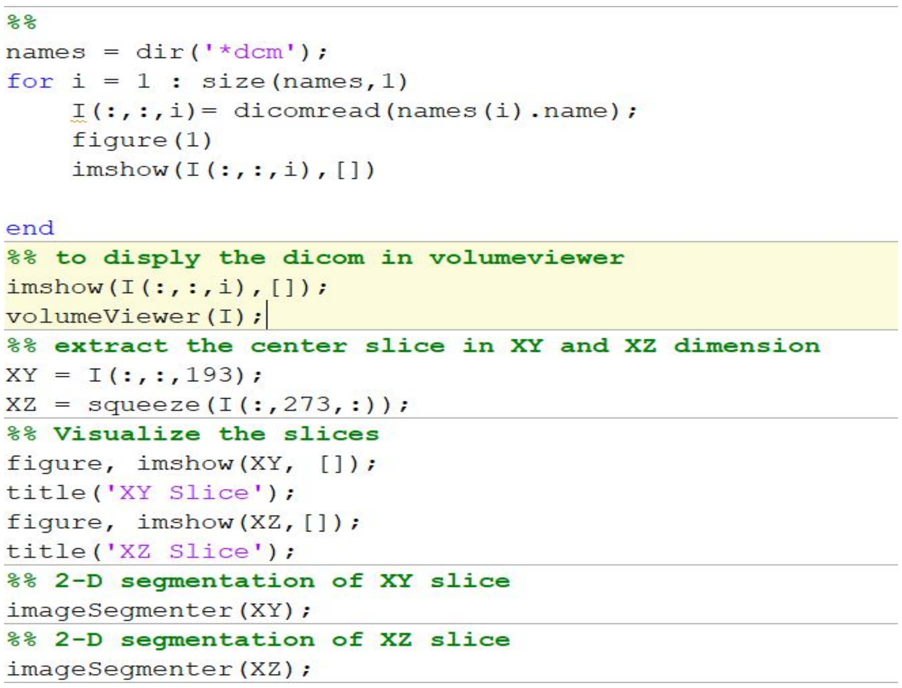


Figure 18: 3D rendering of spine

For this demonstration the spine was rendered as the team is still learning how to identify aneurysms in the DICOM files.



```
%%
names = dir('*dcm');
for i = 1 : size(names,1)
    I(:,:,i)= dicomread(names(i).name);
    figure(1)
    imshow(I(:,:,i),[])

end
%% to disply the dicom in volumeviewer
imshow(I(:,:,i),[]);
volumeViewer(I);
%% extract the center slice in XY and XZ dimension
XY = I(:,:,193);
XZ = squeeze(I(:,273,:));
%% Visualize the slices
figure, imshow(XY, []);
title('XY Slice');
figure, imshow(XZ,[]);
title('XZ Slice');
%% 2-D segmentation of XY slice
imageSegmenter(XY);
%% 2-D segmentation of XZ slice
imageSegmenter(XZ);
```

Figure 19 : Codes to run the DICOM file

The above figure shows all codes that have been used to run the file in the programm.