Implantable Ultrasound Final Report

Group 4: Simon Liu, Dante Navarro, Jessica Su

Technical Summary

Background

According to the National Cancer Institute's Surveillance, Epidemiology, and End Results (SEER) Program, an estimated 168,494 people are living with brain and other nervous system cancers in the United States with approximately 19,200 patients diagnosed every year.[1] Glioblastoma multiforme (GBM) is a fast-growing glioma and accounts for 52% of primary brain tumors and 17% of all brain tumors.[2] The current treatment for GBMs is surgery, followed by radiation and chemotherapy. The surgery is aimed at removing as much of the tumor as possible without injuring the surrounding tissue, but since GBMs are surrounded by a layer of migrating tumor cells, complete removal of the cells is impossible. Surgeons usually perform a craniotomy, which involves drilling three 14mm burr holes, to reach the tumor and remove it. It is crucial to continuously monitor these patients post surgery. The current standard of postoperative monitoring for patients who have undergone neurosurgery is a periodic MRI scan approximately every 3 months.

Problem

Post operative monitoring through periodic MRI scans poses a number of issues for ensuring proper patient recovery. Due to the high costs associated with an MRI scan, insurance plans only cover quarterly scans.[3] Often, this length of time is too long to detect and prevent tumor regrowth, which explains why the prognosis for GBMs, 14 months post-op, has remained stagnant over decades.

Approach

Ultrasound imaging is a strong alternative when it comes to brain imaging. Aside from eliminating radiation exposure and high medical costs, ultrasound images have also been proven to allow radiologists to see the same important neurological structures that an MRI would at a slightly lower resolution. By using ultrasound imaging, we can lower costs significantly while increasing the rate of imaging and maintaining good quality images.[4]

We propose an implantable ultrasound device for long-term post-neurosurgical monitoring in glioblastoma patients designed to fit in a 14mm drilled hole that is already created during neurosurgery. The approach can be split up into four individual components: the ultrasound, microprocessor, wirelessly charged battery, and mobile application.

To begin this project, the primary aim of the Computer Integrated Surgery II course is to develop an iOS app that can monitor the potential regrowth of brain tumors by processing raw ultrasound data into images. The app will allow physicians to monitor bleeding, cysts, and growing tumors through images and timelapses processed from the transmitted data.

Image Processing

The image processing in MATLAB was developed based on the Field II library. WIthin their documentation online, the developers provided examples for simulating a cyst phantom and recreating the image. This was used as a basis of the MATLAB code provided in our private Github repository. The simulation script was developed by combining several scripts provided in the example files. At the very end, minor modifications were made to change the filename and variables saved. A script for converting a mat file to a csv file was also developed for the image processing code in C. Using the image reconstruction code from the examples, minor modifications were made for loading in data, and this was used as a basis for developing the C code as well as a "ground truth" comparison for what the image should look like.

A clinical concern was the computational power draw this might require. However, since the image reconstruction is performed on the mobile app, the power draw does not affect the ultrasound device. A theoretical power consumption study was performed to determine what simulation parameter could affect the power draw of the implanted device. The hypothesis was that changing the resolution of the image (through either the simulation or reconstruction side) would affect the computational time (less time for producing a lower resolution image).

The image reconstruction code in C is based on a FFTW library for calculating the discrete Fourier transform. Aside from this library, all of the other functions (like filtering, etc.) were developed from scratch, some based on existing implementations found online (cited in our technical software documentation) with modifications. A high level description of the image processing is that the envelope of each radiofrequency line, or each row of the simulated data, is extracted and sampled at a frequency of 10 for processing by upsampling and filtering with a FIR filter to create the interpolated image.That image is then resized using bilinear interpolation for an appropriately scaled image based on SI units, not pixels, and then color corrected using a gamma correction computation to alter the luminance to match with the MATLAB image. For further details on the code, please refer to the Software documentation.

Within the app, this image creation code is directly called for processing data from a selected file and saving the color-corrected image into a local directory for display.

App Development

We started off the application development with a single page application which we would then build on. The home screen was created utilizing a simple mockup logo that was provided to us from our mentor, and then three buttons that would then link to another portion of the application when they were created.

Each screen was then developed with the intended use outlined in text to ensure the basic user interface (UI) structure was acceptable to the clinician. The buttons on the home screen now had destinations and were linked in order to navigate from the home screen, to each of the intended use screens, and back. Our main focus of the semester was to ensure successful and robust local raw data manipulation and visualization. We started with the local files because once this workflow was established, receiving the data in the other modalities, specifically bluetooth, can use the same pipeline.

To begin this, we established how to select files that are housed in the native files app on the iPhone. This allows for our image processing code to run on any of the files in this native app. After selection of the single file, we then utilized this data path to run the image processing code which outputs

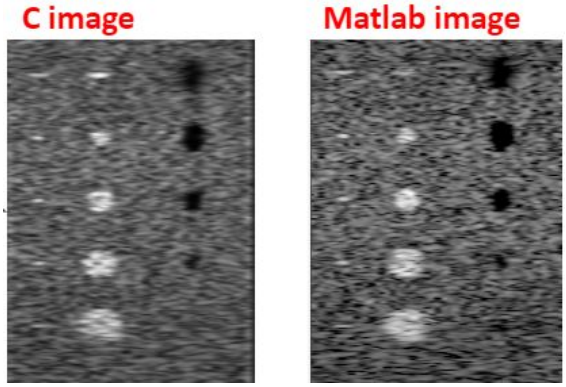
an image. Said image is then visualized on the screen. Since the process is not immediate, an activity indicator was superimposed on the default image until the new image is produced. Once this process was solidified, the next step was selecting multiple raw data files and then outputting several images. By recommendation from our clinician, we added a GIF view that was formulated by these images so that there is a timelapse of multiple images which could provide more insight than a group of individual static images.

Results

Image Processing

In the power consumption study, it was found that the sampling frequency has a tremendous effect on both image resolution and computational time. At a sampling frequency of 10, the current one chosen, the computational time for reconstructing the image was 0.7 seconds. At a sampling frequency of 50, the computational time was 0.22. At a sampling frequency of 70, the computational time was 0.20. The decrease in computational time resulting from an increase in sampling frequency is reasonable because increasing the sampling frequency decreases the amount of data used from the original signal. Since this parameter is also used in the simulation side for a rough processing of the original full signal, changing this parameter on the ultrasound device side should also affect the time required in the same manner.

The MATLAB and C code produced readable ultrasound images from the simulated data.

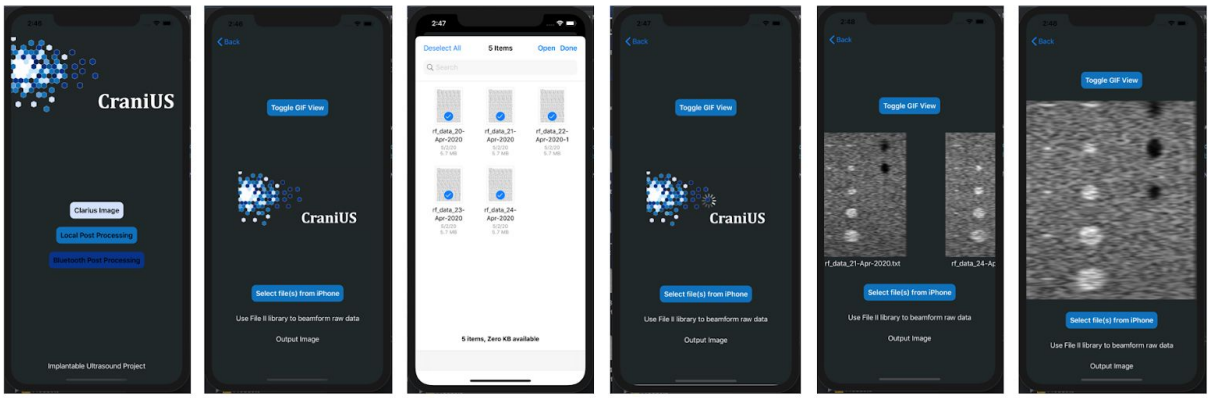


The main observable difference is that the C image looks slightly smoothed. This is a result of the bilinear interpolation used to resize the image to display with appropriate SI unit scaling. For the MATLAB image, that was produced using the native image() function that can accept x and y-axis scaling vectors so in effect that image was not smoothed. Future steps would include applying a sharpening filter to increase the sharpness of the C image.

App Development

We successfully delivered an application that can select multiple local files and display the individual images and a GIF that cycles through the images for easier detection of minute differences.

Below is the workflow of the application:



The initial Home Screen of the Application

Local Post Processing Screen after button is selected

Once the select file button is pressed, the application opens the files view in order to select files

Once files are selected the activity monitor is superimposed on the default picture until the image processing is finished and images are outputted

The images are then displayed in a scroll view with the labels from the associated data file name

Once the GIF Toggle button is pressed, a larger image is shown that iterates through all images to display a time lapse of any changes

A video demo of this workflow can be found on our wiki page.

Significance

The mobile app is a crucial component of the implantable ultrasound system. Without the app, the surgeon has no way of viewing the images taken by an implantable ultrasound device as proposed. Through the results that shown above, with the image processing and user interface that was developed based on the suggestions of the clinical mentor, we have established the clinical workflow of how this app would be used in a clinical setting to monitor a neurosurgical patient in need of frequent monitoring.

Management Summary

Who Did What

Jessica worked on the ultrasound part of the project and preliminary bluetooth-arduino to phone communication. The ultrasound part included simulating ultrasound data using the Field II library in MATLAB and writing the image processing code in MATLAB and C, with integration within the app in Swift. Dante worked on the app development part of the project. This included the user interface, displaying the image and buttons, and allowing the swiping for multiple images as well as the gif option of viewing the rapid success of images like a low resolution video. Simon worked on the Clarius software development part of our original deliverables. This included learning the existing code and relationships between each function call.

Discuss what was planned vs. what was accomplished

The original plan for the minimum deliverable was to establish communication of files between an Arduino-Bluetooth-SD Card Module setup to a phone, which was successfully achieved prior to COVID-19. The original plan for the expected deliverable was to develop a mobile app that communicated with a Clarius probe using their API. This was not accomplished because although code was developed, we found out very late that the probe we had was not a Clarius probe, resulting in a sudden unresolved hardware dependency that we were not able to resolve in time. The original plan for the maximum deliverable was to process simulated ultrasound data using the mobile app, both locally and through bluetooth communication (sending the ultrasound data from an Arduino setup to the phone). We successfully achieved the local image processing via our mobile app. However, we initially planned to use the same Arduino setup as our minimum deliverable. Due to COVID-19, right before JHU closed, the hardware was given to Dante for reproducing the results Jessica obtained in the early weeks so that he could work on the Bluetooth-mobile app communication with our own app, not the Bluetooth terminal. However, there were several code uploading problems that we were collectively unable to resolve, so a new Arduino module was ordered. As of now, the hardware still hasn't arrived, so we will not be able to achieve the entirety of the maximum deliverable. Instead, with the remaining time we had, we worked with our clinical mentor, Dr. Gordon, to refine the app and add functions to improve usability like adding the gif option, which will be very helpful in detecting changes in each successive scan.

Discuss what might be next

We have plans to continue working on the project with varying levels of involvement. Jessica and Simon will continue on the project during the summer, and are looking to continue beyond as well. Dante will be able to lightly assist this summer, with his expertise in app development.

What you learned

We each learned a lot of different skills throughout the project. Jessica learned how to use the Field II library, coding in C, compiling libraries from scratch for iOS development, and basic app development in Swift. Dante learned the new syntax in Xcode called SwiftUI that is unique to the higher iOS development that is currently the suggestion for user interface from Apple. Simon learned how to use the Clarius API in C and deploy external C libraries in Xcode.

Technical Appendix

The code can be found in a private GitHub repository that is linked on the CIIS website. Please email a member of the team for access.

The user manual and documentation of each developed component (software and basic hardware) is linked on the CIIS website.

References

1. Cancer of the Brain and Other Nervous System - Cancer Stat Facts. (n.d.). Retrieved from https://seer.cancer.gov/statfacts/html/brain.html
2. Glioblastoma Multiforme. (n.d.). Retrieved from https://www.aans.org/en/Patients/Neurosurgical-Conditions-and-Treatments/Glioblastoma-Multiforme
3. Glover, L. (2014, September 10). Why Your MRI or CT Scan Costs An Arm and a Leg. Retrieved from https://www.nerdwallet.com/blog/health/mri-ct-scan-costs-arm-leg/
4. Picone, O., Simon, I., Benachi, A., Brunelle, F. and Sonigo, P. (2008), Comparison between ultrasound and magnetic resonance imaging in assessment of fetal cytomegalovirus infection. Prenat. Diagn., 28: 753-758. doi:10.1002/pd.2037