MERIT.jl

The MicrowavE Radar-based Imaging Toolbox: Julia's Version

Aaron Dinesh

January 2024

Introduction

Microwave imaging has started to garner an interest amongst the medical community as an alternative and safer form of imaging when compared to more traditional methods such as X-rays. Recent clinical trials such as the “MARIA M4” system have proven that such microwave-based methods are more comfortable and offer a viable alternative to X-rays [1]. Mammography is not the only area where this imaging modality is being trialed. It is seeing use in areas such as traumatic brain injury detection, bone degradation, and tumor detection [2]. While the hardware has proved effective, the software side leaves a lot to be desired. Researchers often have to code their own data processing pipeline in order to get usable results for their studies. All this development subtracts from time that could be spent developing new algorithms and higher resolution systems. In their haste to complete a paper, bugs could be inadvertently introduced into the code, at best slowing down research while the bug is fixed, while at worst, it could bias the results without the researchers knowing. As such, good software needs to be built that would allow researchers to worry more about designing and testing algorithms rather than worrying about how to code the supporting functions that would allow them to test the effectiveness of these algorithms.

Problem Statement

There are 3 main pillars for any good software framework. It needs to be robust, easy to use and expand, and most importantly, open source. One additional requirement for this library is that it must run on a GPU. This allows for the processing of vast amounts of data, opening the space to more complex algorithms and higher resolution scans. The MERIT toolbox for MATLAB, written by Prof. Declan O’Loughlin et al, has laid the foundation on which other researchers can test imaging software as well as new algorithms. This library has already been used by multiple publications over the years with 13 citations attributed to the original paper and 16 forks of the repository. While MATLAB is easy to use and understand, it can be quite slow to execute. In order to write performant code, another language needs to be considered. Aruoba and Fernandez-Villaverde found, through testing, that MATLAB runs 3x slower than the C++ implementation and around 2.24x slower than the Julia implementation [3]. Julia offers numerous benefits over both C++ and MATLAB. It can be considered a high-level language like MATLAB, while also offering a speed comparable to C++ due to the compilation of the code at run time. Various language features such as multiple dispatch also allow for easy extension of the code to accommodate for nuances in signal processing algorithms that may arise when dealing with scans from different parts of the body. Julia also offers a GPU compiler which allows Julia code to run on the GPU, allowing us to create function implementations that can use the full breadth of computing resources available.

Goals and Objectives

The goal of this library is as follows:

- Create a comprehensive and easy to use microwave imaging library
- Create a data structure and a type-heirarchy that allow for easy extension and flexibility
- Code the functions to run on the GPU
- Publish the library to the Julia packages repository

The stretch goals for this project include:

- Extending the functionality past what the MATLAB library offers
- Implementing more signal processing algorithms for microwave imaging

Tasks and Timelines

ID	Task	Deadline
1	**Create an image from the scans given**	**26 January 2024**
1.1	Implement code to generate a hemisphere	22 January 2024
1.2	Implement code that calculates the delays from every point to the antenna	23 January 2024
1.3	Implement the Delay-and-Sum (DAS) beamformer	25 January 2024
1.4	Create a visualizer to show the resulting image	26 January 2024
2	**Implement the other beamformers**	**2 February 2024**
2.1	Implement the Modified-Delay-and-Sum (MDAS) beamformer (called CDAS in the MATLAB library)	31 January 2024
2.2	Implement the Delay-Multiply-and-Sum (DMAS) beamformer	2 February 2024
3	**Implement the metrics from the MATLAB library**	**9 February 2024**
3.1	Implement the Full Width Half Max metric	5 February 2024
3.2	Implement the Signal-to-Mean Ratio (SMR) metric	7 February 2024
3.3	Implement the Signal-to-Clutter Ratio (SCR) metric	9 February 2024
4	**Make the pipeline customizable and allow for GPU execution**	**23 February 2024**
4.1	Standardise the input and output for a "beamformer algorithm" type function	13 February 2024
4.2	Allow the beamformer function to accept function handles to beamformer algorithms	16 February 2024
4.3	Parallelize the code so it makes effective use of a GPU if present	23 February 2024
5	**End-to-End debug testing**	**1 March 2024**
5.1	Create an end-to-end testing scheme	25 February 2024
5.2	Collect the relevant data to verify the test results	1 March 2024
5.3	Test a part of the library	1 March 2024
5.4	Analyse results from the test	1 March 2024
5.5	Fix bugs according to result analysis. Do 5.2 - 5.5 until all areas have been tested	1 March 2024
6	**Presentation**	**18 March 2024**
6.1	Collect images to put into the presentation	4 March 2024
6.2	Create a programme for the presentation	5 March 2024
6.3	Finalize the presentation and get approval from supervisor	8 March 2024
6.4	Present the presentation	18 March 2024
7	**Final Report**	**12 April 2024**
7.1	Create a LaTeX template for the report	11 March 2024
7.2	Create the first draft	18 March 2024
7.3	Review by supervisor	21 March 2024
7.4	Apply changes suggested to produce a final draft	24 March 2024

The above table outlines the tasks and their deadlines that need to be completed in order for the project and the library to be considered a success. The deadlines set were generous to account for any unforeseen events that may arise that would cause the project to slip. While the probability of such an event occurring is low, measures must still be put in place to safeguard the project in the unlikely event that a delay occurs. As well as generous deadlines, I have also included 14 days of slack at the end, assuming a deadline of April 12, 2024. This again was to account for any unforeseen complications that may arise during development.

Risks and Mitigations

As with any project, there are numerous risks with varying severities that can impact the timeline of the project and the quality of the end result. A good project plan not only identifies and quantifies the risks, but also includes mitigations that can reduce the impact of these risks. These risks are graded on a 5-point scale with the likelihood values being Rare, Unlikely, Moderate, Likely and Certain (1-5 respectively). The impact is also graded on a 5-point scale with the values being Insignificant, Minor, Significant, Major and Severe (1-5 respectively). The product of the severity and likelihood values produces a risk score that can be used to determine how impactful a particular risk can be if it occurs and is not handled appropriately

Risk Description	**Severity**	**Likelihood**	**Risk Score**	**Mitigation**
Missing deadlines	3	2	6	Generous deadlines and inclusion of 14 days of slack at the end
Failing to implement beamformers	3	1	3	There are plenty of resources and reference implementations available online
Failing to run the code on the GPU	2	3	6	There are many workshops online that demonstrate how to write performant Julia code for GPUs
Don't have enough time to perform End-to-End Testing	4	2	8	Perform testing as the code is being written. So that at the end only minimal testing is required
Don't get the final report done in time	5	2	10	The slack days included means that I should still have time to write the report even if deadlines slip
Fail to implement the Type Hierarchy	4	3	12	Conceptually establish the type hierarchy early on in the development process. Look at other libraries that have had to implement a similar hierarchy and consider how they achieved it

References

[1] A. W. Preece, I. Craddock, M. Shere, L. Jones, and H. L. Winton, "Maria m4: clinical evaluation of a prototype ultrawideband radar scanner for breast cancer detection," *J Med Imaging (Bellingham)*, vol. 3, no. 3, p. 033502, 2016. 2329-4310 Preece, Alan W Craddock, Ian Shere, Mike Jones, Lyn Winton, Helen L Journal Article United States 2016/07/23 J Med Imaging (Bellingham). 2016 Jul;3(3):033502. doi: 10.1117/1.JMI.3.3.033502. Epub 2016 Jul 20.

[2] N. Alsbou, K. Cisse, C. Cox, M. Akinbola, and I. Ali, "Medical imaging system design using microwave antennas and portable platform," in *2023 IEEE 17th International Symposium on Medical Information and Communication Technology (ISMICT)*, pp. 1–5.

[3] J. Aruoba, S. Boragan; Fernandez-Villaverde, "A comparison of programming languages in economics: An update," p. 4, 2018.