FASTER SUBPIXEL REGISTRATION FOR FORWARD LOOKING SONAR RECONSTRUCTION[*]

RILEY MARTELL[†]

Abstract. This paper presents a benchmarking comparison of subpixel image registration in Julia and Matlab as applied to forward looking SONAR reconstruction. The phase-correlation area-based registration algorithm is investigated and a novel Julia modification to the existing algorithm for GPU usage is offered. All implementations achieve the same results with varying execution times. Julia consistently outperforms the Matlab counterparts. Results show a 2.97 times speedup for serialized implementations and a 13.56 times speedup for parallel implementations, using Julia as compared to Matlab. The novel Julia implementation utilizing GPUs shows a 6.52 times speedup compared to the serial Julia code.

Key words. Julia, Matlab, SONAR, Image Registration, Phase Correlation

1. Introduction.

Recent advances in forward-looking SONAR technologies lend to large collections of acoustic imagery at high frame rates, providing a useful basis for reconstructing underwater environments. SONAR provides unique challenges compared to optical imagery, such as low signal-to-noise ratios and inhomogenous intensity across the image. Reconstruction methods for all imagery rely on image registration, which is the ability to align two images by means of a model and data transformation. To achieve a full reconstruction of a high frame rate dataset, it would be simple to transform nearby frames via image registration and concatenate for a global reconstruction. However, cumulative errors arise so global alignment is needed to employ consistency between consecutive and non-consecutive image pairs. Many methods exist for global image alignment, one such method that has shown great performance on SONAR datasets is pose-graph optimization. [5] [4]

The full pose-graph optimization for forward-looking SONAR reconstruction is described in [4], and is outside of this project's scope. A short summary suffices to motivate the work.

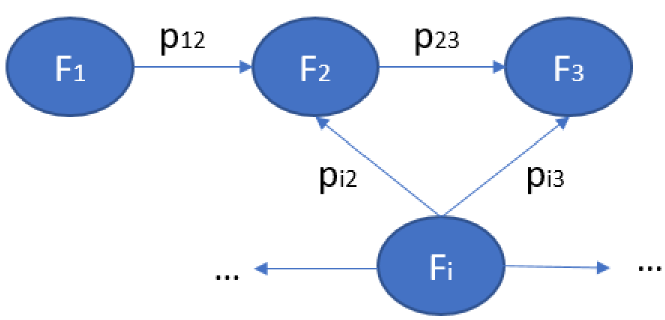

FIG. 1. *Pose-graph structure for forward looking SONAR reconstruction algorithms.*

Suppose you have a dataset of SONAR frames $\mathbf{F} = [F_1, F_2, F_3, ..., F_n]$ and con-

[*]Submitted on 5/16/2023.

[†]MIT Lincoln Laboratory (remartell98@gmail.com).

straints based off neighbor pairwise image registrations $\mathbf{p} = [p_{12}, p_{23}, ..., p_{1n}, p_{2n}, ...]$. The resulting graph structure is shown in Fig. (1). Each vertex of the graph represents the position of the SONAR image on the reference frame. Image registration constraints make up the edges linking corresponding vertices. [5] [4]

The graph is initialized using the simple concatenation of transformations for neighbor vertices. Then, the graph is optimized using a select list of candidate frames for further image registration. One option for selecting candidate frames is to register all frames with the full rest of the set. However, this brings a huge computational burden and becomes untenable very quickly for sequential image registration. The standard approach, set by [5], is to compute registrations of each frame with several neighbor frames using a mixed window. The window size is estimated according operational parameters of the SONAR, such as range and mean velocity. [5] [4]

In practice, this results in a small loop of image registrations for one vertex with its selected neighbors, computed for all vertices. This is still very computationally intensive. For example, one case of image registration using a dataset of 700 frames resulted in over 12,600 calls to the image registration function. Depending on your implementation, this can easily create a bottleneck.

The aim of this project is to investigate options to ease this computational burden. The main contributions of this work are twofold: a series of benchmarks comparing phase-correlation subpixel registration in Julia and Matlab and a novel modification to existing phase-correlation subpixel registration in Julia that utilizes GPUs. All Matlab and Julia code make use of existing implementations [1] [6].

The paper is organized as follows. Image registration is described in section 2 along with a summary of the current implementations in Julia and Matlab. Our benchmarking results are in section 3 followed by discussion. Our new modification to the existing Julia subpixel registration is described in section 5 and the conclusions are described in section 6.

2. Phase-correlation Subpixel Registration Algorithm. Aligning images by means of a model and data transformation, or image registration, has wide applicability yet is often computationally and data intensive [2]. Registration techniques are broadly studied and generally fall into two categories: feature-based approaches and area-based approaches. To estimate the projection relating one image to another, feature-based approaches rely on a small set of localized, distinguishable points while area-based approaches utilize the intensity information from the full image [7],[5]. Area-based methods therefore become advantageous when features are not well localized or have poor resolution, which is typical of forward looking SONAR imagery [7]. Prior work shows sub-pixel accuracy area-based methods outperform similar feature-based methods for simplified forward-looking SONAR geometries, and are thus the focus of this work [4].

Area-based methods, often referred to as Fourier-based methods, assess a similarity metric between two images in the frequency domain. A common similarity metric is the cross correlation. The particular area-based method shown to perform well on SONAR imagery is the phase-correlation registration algorithm, and will be the subject of analysis for this project. [5]

The basis of the phase correlation algorithm is the Fourier shift property, which says that if you have two images, $f(x, y)$ and $g(x, y)$ related by a translation,

$$f(x, y) = g(x - t_x, y - t_y)$$

,

their resulting spectrum (obtained via Fourier transform) encodes this information into the phase

$$F(u,v) = G(u,v)\exp -i(u * t_x - v * t_y).$$

The basic workflow for pixel level accuracy phase-correlation registration is shown in figure 2 and provides the basis for the subpixel level accuracy algorithm. The final translation is achieved through identifying the peak of the cross-power spectrum's inverse Fourier transform.

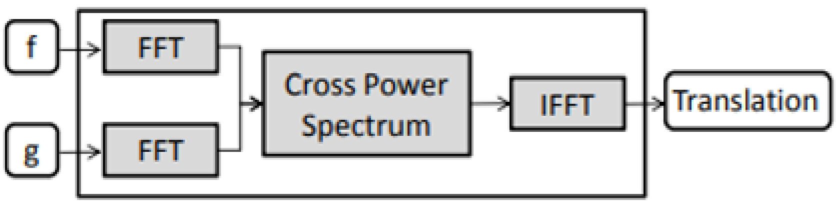

FIG. 2. *Workflow for pixel-level accuracy phase-correlation registration, pulled from [5].*

To achieve subpixel level accuracy in an efficient manor, the following workflow is implemented, developed in [2]. Define $\kappa = 1/s$ where s is the fraction of a pixel the images should be registered within (i.e., s is the subpixel precision). Define M,N as the x and y pixel dimensions, respectively.

- Obtain initial estimate as starting point.
  1. Set $\kappa_0 = 2$
  2. Compute $F(u,v)$ and $G(u,v)$
  3. Embed the product $F(u,v) * G(u,v)$ in a larger array of zeros with dimensions $[\kappa_0 M, \kappa_0 N]$
  4. Compute an inverse FFT to obtain the upsampled cross power spectrum
  5. Locate the peak
- Obtain refined estimate using a 1.5×1.5 pixel region about the initial estimate (in original pixel units).
  1. Set $\kappa_1 \approx \sqrt(\kappa)$
  2. Compute $F(u,v)$ and $G(u,v)$
  3. Embed the product $F(u,v) * G(u,v)$ in a larger array of zeros with dimensions $[\kappa_1 M, \kappa_1 N]$
  4. Compute an inverse FFT to obtain the upsampled cross power spectrum
  5. Locate the peak
- Obtain refined estimate using a $\frac{3}{\kappa_1} \times \frac{3}{\kappa_1}$ region about the new estimate (in original pixel units).
  1. Set $\kappa = \frac{1}{s}$
  2. Compute $F(u,v)$ and $G(u,v)$
  3. Embed the product $F(u,v) * G(u,v)$ in a larger array of zeros with dimensions $[\kappa M, \kappa N]$

4. Compute an inverse FFT to obtain the upsampled cross power spectrum
5. Locate the peak

Typically this phase-correlation registration method is advertised as computationally efficient and less memory-intensive. However, many implementations do not utilize modern computing features such as parallelism or graphical processing units (GPUs) [4],[1]. A high-performance solution in Julia has been developed and assessed for feature-based registration algorithms as applied to medical imagery, where they often perform very well [3]. Performance of area-based algorithms is less documented, and no studies seek a high-performance solution for SONAR data to the author's awareness.

3. Benchmarking results. Here we state the first results, a comparison of the above phase-correlation subpixel registration implemented in Matlab and Julia [1], [6]. Corresponding to the motivation use case, phase-correlation registrations were performed on real-world SONAR data collected by an unmanned underwater vehicle for mapping purposes.

Both implementations computed the same results within a tolerance of $\epsilon = 0.001$, deemed acceptable for this use case. Each frame has dimensions [661X484]. A subset of 9 frames were selected for registration within a loop, and execution time is evaluated for the entire loop. This directly applies to the pose-graph optimization for SONAR reconstruction application, we seek to optimize for.

Figure 3 shows the results for a serialized loop using the existing Matlab and Julia methods. The loop was performed 10 times to ascertain confidence in the results. Matlab timing is computed using the 'tic' and 'toc' functions. Julia timing is generated using the '@btime' function from the BenchmarkTools package.

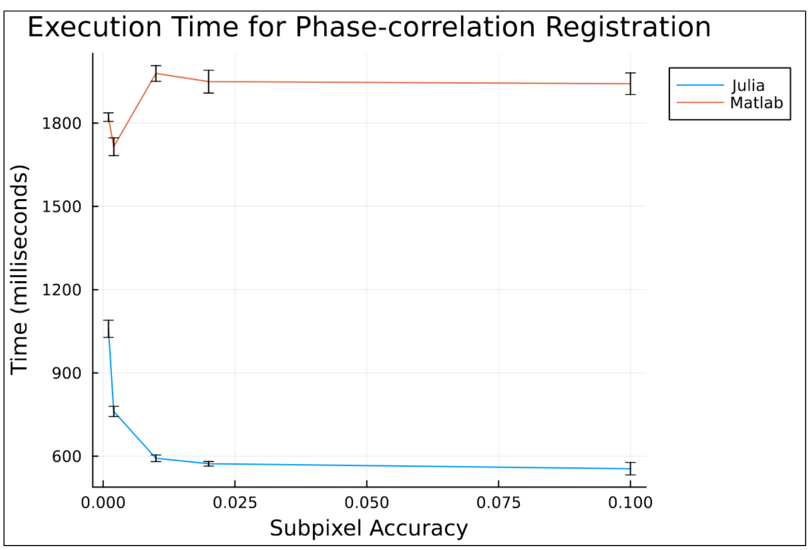

FIG. 3. *Baseline comparisons for phase-correlation registration implementations.*

Figure 4 shows the results for a parallelized loop using the existing Matlab and Julia methods. Parallelism in Matlab is achieved using the 'parfor' technique. Parallelism in Julia is achieved using the Threads package. Both were performed on an 8 core machine with Intel Xeon E5 CPUs. The loop was performed 10 times to ascertain confidence in the results. Matlab timing is computed using the 'tic' and 'toc' functions. Julia timing is generated using the '@btime' function from the BenchmarkTools package.

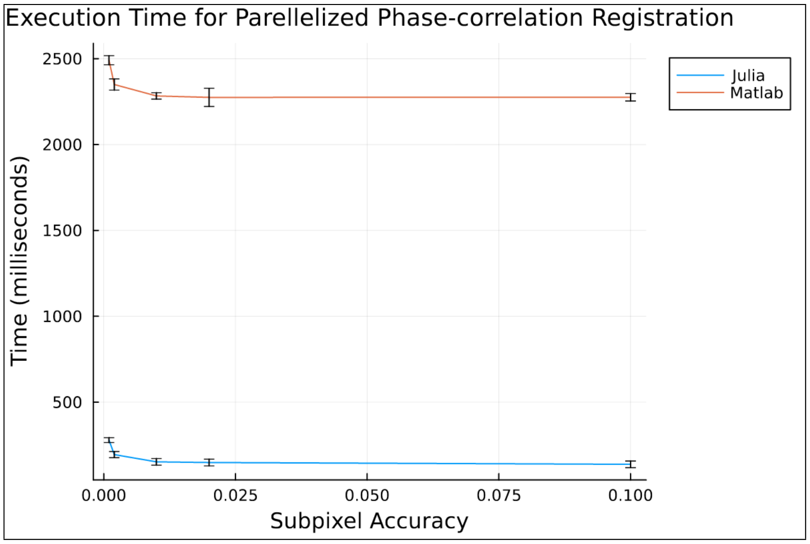

FIG. 4. *Parallelized comparisons for phase-correlation registration implementations.*

4. Discussion of benchmarking results. Julia implementations give much faster execution times for both the serial and parallelized versions as compared to Matlab. This is expected because the Julia code is type stable, meaning its compiled version is essentially statically-typed after the first call. Even though our function is called in a loop, Julia is able to compile and optimize the specific registration function rather than having to optimize for the entire loop in its underlying LLVM code, which is advantageous. Matlab is known to run faster for vectorized code with pre-allocated arrays. While the existing implementation does a good job at pre-allocation, it is not well vectorized, so the execution time is understandably slower.

The parallelized Julia code is much faster than both Matlab versions and the serial Julia version. Despite this loop not being an obvious candidate for optimization due to its small number of iterations, each function call is independent so Julia is able to execute this on multiple threads with little overhead. Matlab, on the other hand, experiences a great deal of overhead and is surprisingly much slower than its serial counterpart. There are no obvious oddities to Matlab's phase-correlation implementation that would cause such a slow down, such as use of parallelization within one of the sub-functions. It is generally advised to assess the computational burden of function calls within a loop before introducing parallelism in Matlab, as less burdensome functions often have too much overhead for a benefit, and this seems to be the case with phase-correlation registration.

Each method scales similarly with respect to the degree of subpixel accuracy. Requesting registration within a smaller fraction of a pixel results in longer execution times. This is due to higher dimension data from upsampling, which is more expensive for each operation.

5. Algorithmic results. Here we describe our second result, a novel modification to the existing subpixel phase-correlation registration in Julia utilizing CUDA to execute expensive operations on GPUs. This algorithm is based on the Subpixel-

Registration package and has been tested with a NVIDIA Volta GPU. This code is not particularly optimized, but has shown beneficial performance over the standard algorithm limited to CPUs.

```julia
function upsampled_dft(data::CuArray{T},region_size,upsample_factor,offsets,) where {T<:Complex}
    shiftrange = CuArray(collect(1:Int64(region_size)))
    idxoffset = map(first, axes(data))
    sample_rate = inv(T(upsample_factor))
    freqs = CuArray(fftfreq(size(data, 2), sample_rate))

    off2 = CuArray([last(offsets)])
    ioff2 = CuArray([last(idxoffset)])

    kernel = @. (shiftrange - off2  - ioff2) * freqs'
    kernel = kernel .* (2*pi)
    kernel = map(x->cis(x), kernel)
    _data = kernel * data'

    freqs = fftfreq(size(data, 1), sample_rate)
    off1 = CuArray([first(offsets)])
    ioff1 = CuArray([first(idxoffset)])

    kernel = @. (shiftrange - off1 - ioff1) * freqs'
    kernel = kernel .* (2*pi)
    kernel = map(cis, kernel)
    _data = kernel * _data'
    return _data
end
```

FIG. 5. *Julia code snippet that utilizes GPUs using CUDA functionality.*

Only most expensive portion of this algorithm, the upsampled discrete Fourier transform, is shown for brevity. Full code will be made available on GitHub at https://github.com/remartell/mit_18337_SubpixelRegistration.git. CuArrays were utilized to implement GPU calculations. Although this comes at a memory cost for transferring data between the CPU and GPU, execution time was the metric of interest for this project.

Figure 6 shows a comparison for each Julia implementation for a variety of sub-pixel accuracy requirements. As expected, the serial version performs worst as there are no optimizations. The parallelized version performs, on average, 3.90 times faster than serial. The CUDA version performs, on average, 1.68 times faster than the parallel version and 6.52 times faster than the serial version.

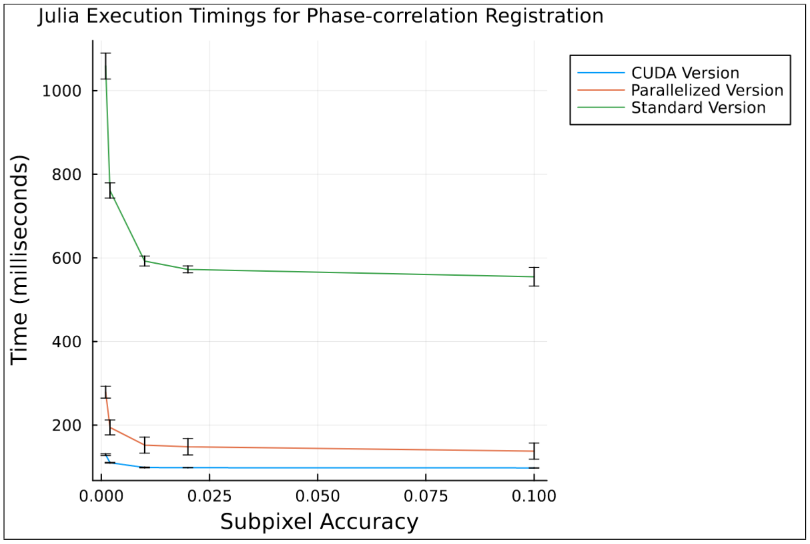

Fig. 6. *Julia comparisons for various phase-correlation registration implementations.*

6. Conclusions. In this work, we assessed performance of the phase-correlation image registration algorithm in Julia and Matlab as used in forward looking SONAR applications. It is well confirmed that Julia outperforms Matlab with the serial and parallel implementations. For serialized implementations, Julia executes 2.97 times faster than its Matlab counterpart. For parallelized implementations, Julia executes 13.56 times faster than its Matlab counterpart. A GPU comparison was not possible due to limitations in Matlab's GPU code conversion tool, but Julia clearly showed benefit for using GPUs with great ease to the user. Compared to the Julia serial version, the CUDA version runs 6.52 times faster.

Following the original motivation of this project, the example of 12,600 function calls to the phase-correlation registration function would take approximately 27 minutes to achieve $\frac{1}{1000}$ subpixel accuracy when utilizing GPUs. Comparatively, Matlab's fastest offering, the serial version, would take over 8 hours to complete these computations. Julia provides a serious benefit with little cost to the user.

Future work on benchmarking would compare the memory use for Matlab and Julia algorithms. Matlab does not have easy command-line tools for assessing the memory footprint like Julia's BenchmarkTools and CUDA packages, so this was not possible for this project. Further work on the novel Julia algorithm would optimize the CUDA implementation, perhaps utilizing a GPU kernel to avoid any CPU calculations.

Acknowledgments. I would like to acknowledge Paul Ryu who assisted in motivating this work and provided feedback throughout.

REFERENCES

[1] M. Guizar, *Efficient subpixel image registration by cross-correlation*, 2023, https://www.mathworks.com/matlabcentral/fileexchange/18401-efficient-subpixel-image-registration-by-cross-correlation.

[2] S. T. M. Guizar-Sicairos and J. R. Fienup, *Efficient subpixel image registration algorithms*, Optics Letters, 33 (2008), pp. 156–158.

[3] S. V. M. Van Gendt, T. Besard and B. D. Sutter, *Productively accelerating positron emission tomography image reconstruction on graphics processing units with julia*, International Journal of High Performance Computing Algorithms, 36 (2022), pp. 320–336.

[4] X. C. Y. P. J. S. N. Hurtos, S. Nagappa, *Evaluation of registration methods on two dimensional forward-looking sonar imagery*, IEEE RSJ International Conference on Intelligent Robots and Systems, (2013).

[5] Y. P. J. S. N. Hurtos, X. Cuf, *Fourier-based registrations for two-dimensional forward-looking sonar image mosaicing*, IEEE RSJ International Conference on Intelligent Robots and Systems, (2012).

[6] romainFr, *Subpixelregistration*, 2016, https://juliapackages.com/p/subpixelregistration. Accessed on 15 03,2023.

[7] Y. X. S. G. e. a. X. Tong, Z. Ye, *Image registration with fourier-based image correlation: A comprehensive review of developments and applications*, Selected Topics in Applied Earth Observations and Remote Sensing, 12 (2019), pp. 4062–4081.