Draft version March 4, 2023
Typeset using LaTeX default style in AASTeX63

ALMA Development Study Program

Beyond Black Hole Images: Extending New Imaging Techniques from the EHT to ALMA

Kazunori Akiyama [1,2,3] Lynn D. Matthews [1] and Leonid Benkevitch [1]

[1] Massachusetts Institute of Technology Haystack Observatory, 99 Millstone Road, Westford, MA 01886, USA
[2] National Astronomical Observatory of Japan, 2-21-1 Osawa, Mitaka, Tokyo 181-8588, Japan
[3] Black Hole Initiative, Harvard University, 20 Garden Street, Cambridge, MA 02138, USA

ABSTRACT

The pursuit of high angular resolution, high-fidelity imaging at millimeter and submillimeter wavelengths is a fundamental goal of ALMA. This ALMA Study project aimed to improve the effective spatial resolution and image fidelity of the current ALMA array and its planned extended array through the use of regularized maximum likelihood (RML) methods that were intensively developed in the community of the Event Horizon Telescope (EHT). As the current RML software packages in use by the EHT are designed for a single-band continuum imaging with a sparse array, they require some modifications to be extensible to the more general imaging needs of ALMA. As a first step we therefore explored the needs of an RML-based software platform dedicated to the requirement of ALMA imaging. We chose the Julia programming language after exploring popular modern high-performance computing frameworks. We developed a software suite in Julia including an implementation of RML Methods. While the Julia RML implementation is yet mostly neither optimized nor parallelized, the current codes already achieve a comparable speed to an optimized multi-CPU-threaded code written in Fortran 90 & C++ with a significant improvement in the image fidelity at a high-dynamic range. All developed packages are publicly released in Julia's central package repository and can be easily installed through Julia's package manager.

1. INTRODUCTION

Meeting the scientific demands for high angular resolution, high-fidelity imaging at millimeter and submillimeter wavelengths serves as a key driver of ongoing ALMA development efforts. The current ALMA development plan (Carpenter et al. 2020) aims to achieve an additional factor of 2 to 3 improvement in angular resolution. This demands either routinely observing at higher frequencies with the current longest baselines, and/or increasing the maximum baselines by a factor of 2 to 3. However, both options are technically challenging for high-fidelity imaging, due to larger calibration errors and/or much less uniform uv-coverage on the longest baselines. An important technical frontier is therefore the development of robust imaging algorithms which can deal with larger calibration uncertainties and effectively use the planned outrigger stations for enhancing the angular resolution and fidelity of ALMA images.

In the last decade, the technical challenges of achieving the first very long baseline interferometry (VLBI) imaging observations at wavelengths as short as $\lambda 1.3$ mm with the Event Horizon Telescope (EHT) have accelerated the development of new imaging techniques, collectively known as regularized maximum likelihood (RML) methods (e.g. Akiyama et al. 2017a,b; Chael et al. 2016, 2018; Event Horizon Telescope Collaboration et al. 2019b). RML methods also have strong potential to overcome and improve current and future challenges of ALMA imaging in three ways: (1) allowing high-fidelity reconstructions, even at the modest super resolution, i.e. 2 3 times finer than that of traditional CLEAN methods(e.g. Högbom 1974; Cornwell 2008); (2) the capability to reconstruct images directly from closure quantities (Chael et al. 2016; Akiyama et al. 2017b; Chael et al. 2018), free from antenna-based calibration errors; and (3) the ability to handle intrinsically multidimensional emission, such as time-variable emission structures (Johnson et al. 2017). The aforementioned advantages of RML methods over CLEAN have now been demonstrated not only for VLBI, but also for single-band continuum observations with ALMA (Chael et al. 2016; Yamaguchi et al. 2020), VLA (e.g. Matthews et al. 2018), and also simulated ngVLA observations (e.g., Akiyama & Matthews 2019). However, since the current RML packages were designed for performing single-band continuum observations with only several antennas, the general application of RML methods to ALMA observations requires modifications to both the software packages and the imaging algorithms.

This ALMA Study project aimed to improve the effective spatial resolution and image fidelity of the current ALMA array and its planned extended array, by extending RML methods from EHT to ALMA. We first explored the software infrastructure suitable for the extension of RML methods to ALMA. In particular, we considered the following aspects:

- **high-performance:** the most popular RML packages in the EHT community are written purely in Python (e.g. `eht-imaging`; Chael et al. 2016, 2018) or in a hybrid of Python and Fortran 90 (e.g. `SMILI`; Akiyama et al. 2017a,b). The current software packages are significantly limited by the performance of Python, and are generally not scalable to the ALMA case (up to 64 antennas), where the data sets may also have large numbers of spectral channels. The software needs to be developed in a high-performance language to remove the existing bottlenecks in the current RML packages.
- **scalability:** large data volume and high dimensionality of ALMA imaging will significantly benefit from high-performance parallel computing scalable to CPU/GPU computer clusters. The software should be developed within a framework that allows for ready extension to parallel and distributed computing.
- **composability and extendability:** the community-wide development of RML or similar forward-imaging methods have been increasingly accelerated given the challenges of next-generation interferometers. The developed software ideally should offer a software infrastructure and tool kits including basic data types and associated common functions to facilitate software reuse and future community software development after this program.
- **installability:** current RML packages (e.g., `SMILI`) often depend on separate community packages (e.g. `FFTW` for the fast Fourier Transform (FFT), an optimized `BLAS` package for accelerated linear algebra computations), which makes the installation process complicated. The developed software, including its dependencies, ideally should be easily installed for users in the wider astronomy community. This demands a mature software ecosystem with a well-maintained package system.

We explored modern high-performance computing frameworks popular in scientific communities, including those offering a significant acceleration for Python, for instance `Google JAX`[1] based on Google's `TensorFlow`[2] and META AI's `pytorch`[3]. Among these, we found the Julia programming language [4] to be a uniquely powerful software framework satisfying all of the above criteria. We developed an open-software suite in Julia including the implementation of RML methods, now available for the community through Julia's central software repository. In this report, we briefly summarize the Julia programming language in Section 2, and the software suite developed under this program in Section 3. The software suite will continue to be developed after this program. We present a summary in Section 4.

2. THE JULIA PROGRAMMING LANGUAGE

The Julia programming language (Bezanson et al. 2017) is a high-level, high-performance programming language designed for numerical and scientific computing, data science, and machine learning. It was first introduced in 2012 and has gained significant popularity in the scientific computing community due to its unique combination of features relevant to the needs for ALMA imaging, including the following:

- **easy to use/write:** similar to Python, Julia is dynamically typed and has good support for interactive use, including its efficient interactive shell `Julia REPL`[5] and `Jupyter Notebook/Lab`[6].
- **fast:** programs written in `Julia` are compiled into efficient native machine code using a just-in-time (JIT) compiler for multiple platforms via the LLVM compiler infrastructure. The dynamically compiled programs have a speed comparable to optimized `C` or `Fortran`, which are often more than a hundred times faster than Python.
- **reproducibility:** Julia offers user-friendly, reproducible environments that allow users to quickly build a local environment for each project, and easily recreate the same Julia environment across different platforms.

[1] https://github.com/google/jax
[2] https://www.tensorflow.org/
[3] https://pytorch.org/
[4] https://julialang.org/
[5] https://docs.julialang.org/en/v1/stdlib/REPL/
[6] https://github.com/JuliaLang/IJulia.jl

- **package manager:** Julia has an efficient package manager similar to Python's pip, which allows users to easily install packages and their dependencies in a few commands.
- **parallel computing:** `Julia` is natively designed for multi-threaded vectorized calculations as well as distributed computing over multiple CPUs and GPUs. It provides built-in primitives for parallelism, which allows users to write high-level parallelized algorithms without having to worry about the low-level details of distributed computing.
- **composability and extendability:** Julia adopts 'multiple dispatch' as a paradigm that encourages composability, allowing for the easy implementation of portable and extendable modules. Julia covers a wide range of programming paradigms, including functional programming, object-oriented programming, and metaprogramming.
- **Interoperability:** Julia can easily interoperate with other programming languages, including Python, C/C++, and Fortran.

Compared to other frameworks, we found that Julia in general provides more flexible handling of multidimensional data without compensating for its speed due to its native design. For instance, `Google JAX` and `PyTorch` have their own data types for multi-dimensional data, which are all "immutable" once data are created in their data types, they can not be changed and need to be recreated for any edit. We found that their more restrictive framework made the implementation of high-dimensional imaging and associated data calibration algorithms more difficult and complicated than Julia.

In addition to the above benefits from the language design, thanks to more than a decade of community development, Julia now has fast, computationally efficient implementations of many key algorithms. Julia has core support for automatic differentiation and many inference algorithms, which are essential to develop the complicated multidimensional imaging algorithms that we envision for ALMA imaging. Another key algorithm for RML and similar imaging methods with closure quantities is the non-uniform fast Fourier transform (NUFFT), allowing a fast, non-equidistant Discrete Fourier Transform from astronomical images to visibilities on non-equispaced coverage in Fourier space. Julia has a fast Julia-native implementation of NUFFT algorithms,[7] (`NFFT.jl`; Knopp et al. 2022) both for CPU and GPU architectures. `NFFT.jl` is one of the fastest NUFFT libraries to date, which significantly outperforms the `C`-based `NFFT3` library (Keiner et al. 2009) used in `eht-imaging` and offers comparable performance to the `C++`-based Flatiron Institute NUFFT (`FINUFFT`; Barnett et al. 2019) library used in `SMILI`, one of the fastest NUFFT libraries publicly available.

The strong potential of Julia has led to the rapid growth of its astronomy community[8]. Julia has a sufficient ecosystem including a wrapper of the `casacore` library required to handle ALMA interferometric data in `CASA`'s Measurement Set. As it has excellent interoperability with Python, it offers efficient use of Python's astronomy packages relevant to ALMA, for instance, `astropy`. `Julia`'s strong potential for radio interferometric data analysis has been demonstrated by the `Comrade` Bayesian imaging/modeling package (Tiede 2022). This package achieves >10 times faster speed than Python-based `eht-imaging` for model fitting and orders of magnitude faster speed than `C++`-based `THEMIS` (Broderick et al. 2020) for Bayesian inference on EHT data due to the availability of more advanced sampling techniques.

For the above reasons, we chose Julia to extend the RML methods for ALMA in this study program. Under this program, we have developed an open-source software suite including the implementation of the RML imaging methods as described in the next Section.

3. RML IMPLEMENTATION IN Julia

3.1. *A Julia Software Suite*

In this study program, we have launched the EHT Julia Organization[9] to develop a modern high performance software suite in Julia that provides high-resolution algorithms and techniques originally developed for the EHT to the broader astronomical community. This program provided a series of several open-source packages, allowing a basic implementation of RML methods including the following:

[7] https://github.com/JuliaMath/NFFT.jl

[8] for instance, http://juliaastro.org/

[9] https://github.com/EHTJulia

`EHTImages.jl`—a Julia package that defines data types and implements basic functions to handle five-dimensional astronomical images for radio interferometry. The package provides features meeting the needs for multi-dimensional imaging including the following ones:

- Native support of the five-dimensional images (x, y, frequency, polarization, time) in a self-descriptive data format.
- Nonequispaced grids in time for the application of dynamic imaging methods (e.g. Bouman et al. 2016; Johnson et al. 2017), and in frequency for the application of multi-frequency imaging methods (e.g. Chael et al. 2022).
- Interactive tools to edit, plot, analyze and transform images through pure Julia native functions
- Support for multiple data formats for loading and writing, including a FITS format compatible with `CASA`, `AIPS`, and `DIFMAP` software packages, as well as the image data formats of `eht-imaging` and `SMILI` that are more popular in the EHT and VLBI communities.

`EHTUVData.jl`—a Julia package that defines data types and implements basic functions to handle multichannel, multispectral-window visibility data products anticipated by general interferometric data sets, including those from ALMA observations. Under this program, we implemented basic functions that allow loading of interferometric data sets in the popular UVFITS format widely used in community software packages, including `CASA`, `AIPS`, and `DIFMAP`.

`RMLImaging.jl`—a Julia package that provides a software tool kit of RML imaging methods of radio interferometry. The current version focuses on the implementation of RML imaging methods for high-angular-resolution imaging within a modest field-of-view where the direction-dependent effects are ignorable. The tool kits and algorithms are designed to meet the need for multi-dimensional imaging with millimeter interferometry (e.g. ALMA) and VLBI (e.g. EHT). The library aims to provide the following features:

- Julia native implementation of the end-to-end RML imaging process: all codes involved in RML imaging methods are written in Julia. This allows for a significant acceleration of popular Python implementations of RML imaging methods. The Julia-native codes allow the utilization of various powerful packages in the Julia's ecosystem.
- Composable tool kits: a set of data types and functions for the sky intensity distributions, NUFFT, regularization functions, data likelihood functions, and optimization to solve images. This allows interested users to extend this package and implement their own imaging methods.
- Implementation compatible with automatic differentiation (AD) packages: the relevant functions for imaging (e.g., forward/adjoint transforms between sky images and Fourier-domain visibilities, regularization functions) are all compatible with automatic differentiation (AD) packages in the Julia ecosystem. This allows easier implementation of algorithms using complicated sky models or set of data products, as well as usage of the various inference algorithms in the Julia ecosystem.

Under this program, we have implemented a basic RML method function and associated data types and methods for single-frequency, single-polarization imaging. We will show some example reconstructions in Section 3.2.

All packages developed under this program are publicly available in the software repository of the EHT Julia Organization on GitHub. Most of the packages are released in Julia's central package repository. For instance, `RMLImaging.jl` and its dependencies can be installed in the standard way from Julia's official package manager as follows:

```
using Pkg
Pkg.add("RMLImaging")
```

Users can easily install all software products from this ALMA study program in the same way. Documentation is currently in preparation and will be released in each GitHub repository in the standard way.

3.2. *Example Reconstructions and Performances*

We tested a basic RML method function implemented in `RMLImaging.jl` against data sets from real ALMA observations and synthetic ALMA VLBI observations. The current RML implementation is single-thread and not yet optimized for the speed except for the NUFFT function using `NFFT.jl` which offers native support of multi-CPU-threads parallel computing. We here compare the performance against `SMILI` v0.2.0 which offers one of the fastest

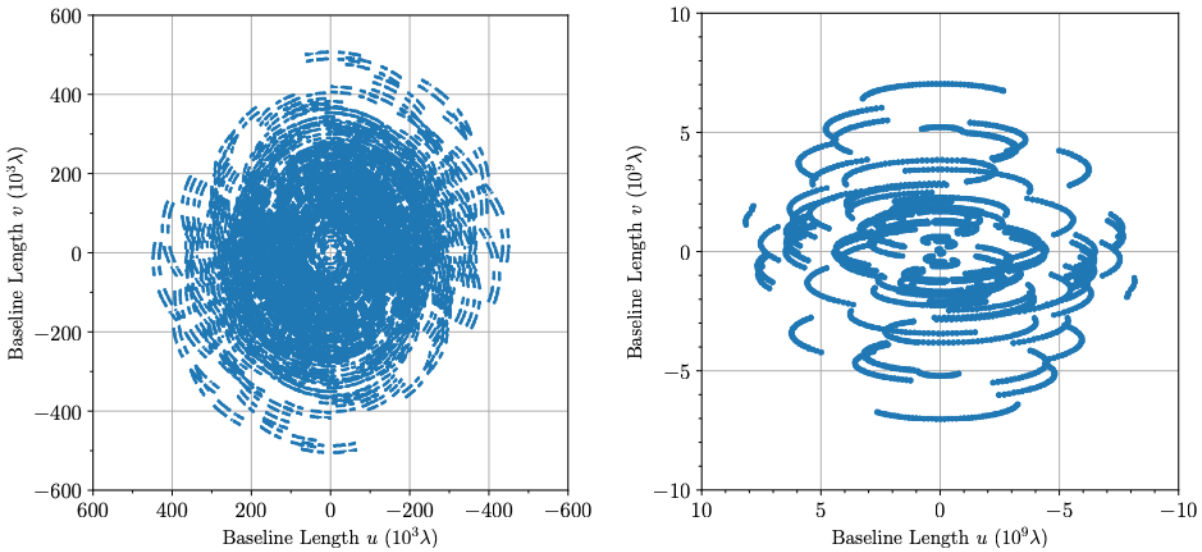

Figure 1. *uv*-coverage of observational data sets used in the examples. (*Left*) real ALMA observations of the protoplanetary disk HD 142527 in Band 7 (330 GHz). We used Data 1 of Yamaguchi et al. (2020), obtained with a compact configuration used by those authors to evaluate the fidelity of super-resolution imaging. The data set used for imaging includes in a total of 223,249 visibility data points. (*Right*) synthetic ALMA VLBI observations of M 87* in Band 6 (230 GHz) with a planned array for the late 2020s with 14 stations. The data set contains 2461 visibilities.

implementations for RML methods written by highly optimized and fully multi-threaded Fortran 90 codes with the usage of multi-threaded C++-based `FINUFFT` library. Although `SMILI` and `RMLImaging.jl` differently implement RML imaging methods — for instance `SMILI` uses bound-constrained L-BFGS-B algorithms for optimizations while `RMLImaging.jl` use unbounded L-BFGS algorithms with a positivity constraint we align the imaging settings as much as possible. We use the same sets of the image fields-of-view and pixel sizes and the equivalent parameters for the regularization functions. L-BFGS iterations are fixed to 5,000 for both imaging software packages. All imaging was performed in a 2019 model of the 16-inch MacBookPro with eight 2.3 GHz CPUs of Intel Core i9.

In Figure 2, we show example reconstructions of an ALMA Band 7 (330 GHz) observational data set of the protoplanetary disk around HD 142527, one of the most well-studied transition disks. We use "Data 1" from Yamaguchi et al. (2020), obtained from a compact configuration and used to evaluate the performance of RML methods for super-resolution imaging against a higher-angular resolution data set at the same observing band (see Yamaguchi et al. 2020, for details). Visibility data are frequency-averaged within each spectral window (spw) and time-averaged with a 60 sec resolution. We use all four spectral windows in the imaging, which provide 225,249 visibilities with the *uv*-coverage shown in Figure 1. The imaging was carried out with the same image pixel size of 0.″05 and field-of-view (5″) as used by Yamaguchi et al. (2020). For the regularization function, we used the total squared validation (TSV) (Kuramochi et al. 2018) with `SMILI` and `RMLImaging.jl`, and also the new TSV-L regularization (Akiyama et al. in prep.) with `RMLImaging.jl`.

With fully parallelized computations over 16 threads, `SMILI` took ∼250 seconds for imaging. On the other hand, `RMLImaging.jl` took ∼500 seconds with a code mostly single-threaded except for the NUFFT part. While `RMLImaging.jl` is not yet optimized or even fully parallelized, the code already achieved a speed comparable to `SMILI`'s Fortran 90/C++-based imaging function.

As shown in Figure 2, we confirmed that `RMLImaging.jl` provides broadly consistent images with `SMILI` for the same regularization functions. The images provide the ring of the outer disk with a diameter of ∼ 2" and also the fainter point-like emission from the inner disk at the center of the outer disk emission. We also show a reconstructed image with a new TSV-L regularization designed for high-dynamic-range astronomy images. The TSV-L regularization, which allows one to denoise and constrain the intensity distributions over the different orders of magnitude, eliminate many compact, faint blob-like artifacts surrounding the outer disk seen in TSV images, and enhance the overall dynamic range of the reconstruction.

To show the performance of the code against much higher dynamic range images with a modestly wide field-of-view, we show in Figure 3 example reconstructions of a full-track synthetic observational data set in Band 6 (230 GHz)

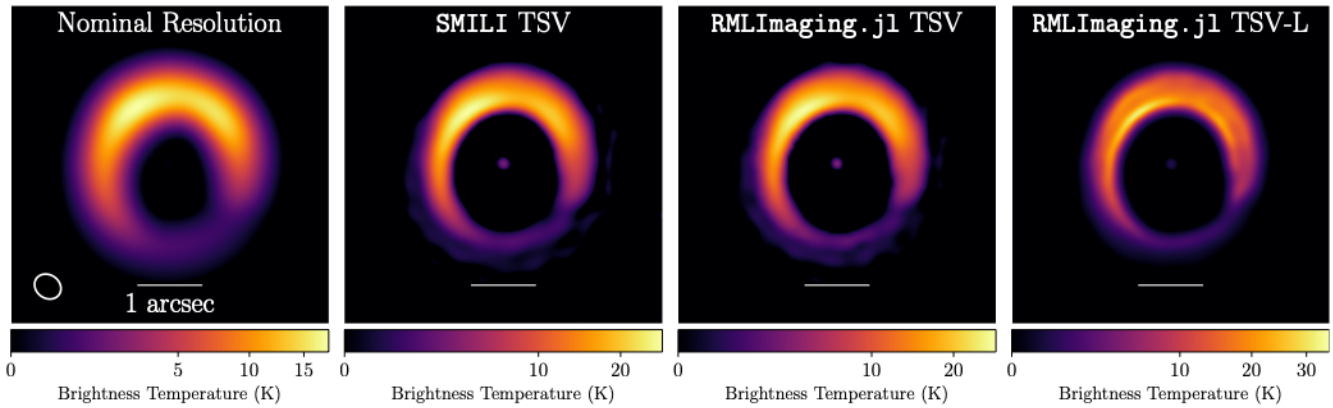

Figure 2. ALMA images of the protoplanetary disk around HD 142527 at 330 GHz, reconstructed with RML imaging methods. The leftmost panel shows a RML reconstruction blurred at the resolution of uniform weighting, to show the image at a nominal resolution. The other three panels show raw (i.e. non-blurred) RML reconstructions; from the left to the right: a `SMILI` reconstruction with TSV regularization (Kuramochi et al. 2018); an `RMLImaging.jl` reconstruction with TSV regularization; an `RMLImaging.jl` reconstruction with the new TSV-L regularization (Akiyama et al. in prep.).

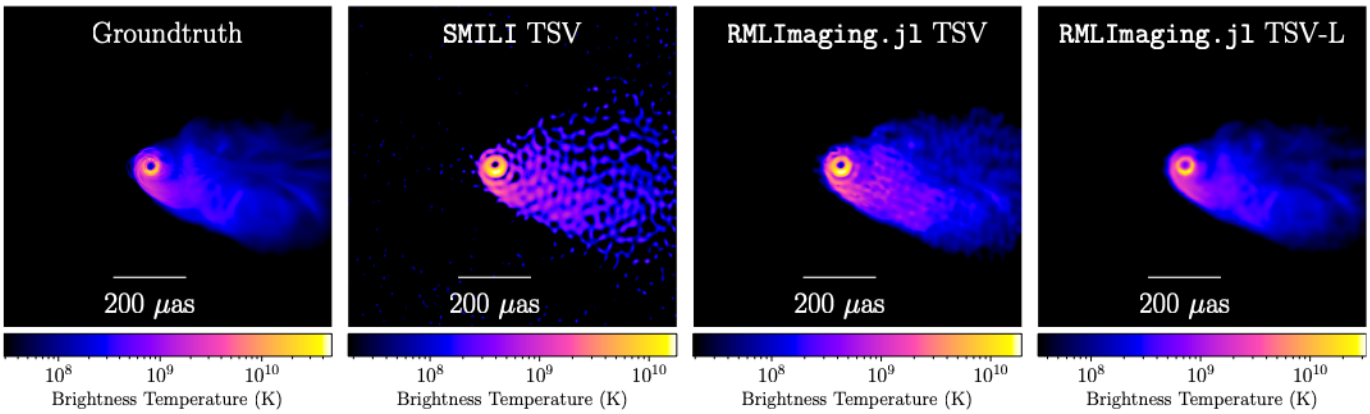

Figure 3. Images from synthetic EHT observations of a GRMHD model for M87* (Chael et al. 2019) at 230 GHz with a planned array for the late 2020s. From the left to the right, we show the groundtruth image, a raw (i.e. non-blurred) `SMILI` reconstruction with TSV regularization, a raw `RMLImaging.jl` reconstruction with TSV regularization and then the raw `RMLImaging.jl` reconstruction with TSV-L regularization.

with a 14-station array of the EHT planned for the late 2020s. We adopt an image from a general relativistic magnetohydrodynamic (GRMHD) model (Chael et al. 2019) as the ground truth. The simulated observations are conducted in `eht-imaging` library, with the inclusion of only thermal noises added into visibilities based on the typical sensitivities of the EHT stations (Event Horizon Telescope Collaboration et al. 2019a). The resultant data set has 2461 visibilities and the *uv*-coverage shown in Figure 1. The images are reconstructed with the same field-of-view of 1024 μas and pixel size of 2 μas, which are exactly the same as the ground truth model.

In Figure 3, we show the RML reconstructions with `SMILI` and `RMLImaging.jl` along with the ground truth image. For this data set, `SMILI` and `RMLImaging.jl` achieved almost the same speed `SMILI` took 250 seconds while `RMLImaging.jl` took 300 seconds. However, as shown in Figure 3, the image fidelity has been significantly improved with `RMLImaging.jl`. Even with the same TSV regularization, the `RMLImaging.jl` reconstruction suppresses blob-like artifacts and provide a higher performance at dynamic ranges of 10–100. The new TSV-L regularization further provided a much-higher fidelity even at a dynamic range of 1000, capturing the filamentary structures seen in the plasma flow of the jet launched from the central black hole. The results demonstrate that the current `RMLImaging.jl` implementation already provides a strong potential to improve the overall fidelity of high-dynamic-range imaging without compromising the computational time.

4. SUMMARY

In this ALMA Study program, we studied pathways to extending RML imaging methods to ALMA. We first explored modern high-performance computing frameworks that allow for (1) high-performance and scalable computing required for ALMA imaging; (2) easy installation and utilization of software tools by the broader community; and (3) facilitating community involvement in software and algorithm development for ALMA data after the conclusion of this program. We identified the Julia programming language as a dedicated framework that satisfies all the above requirements for ALMA imaging. We have developed a software suite for high-resolution radio interferometry with ALMA and other interferometric arrays, including a basic RML imaging method. Tests with real ALMA observational data and also simulated ALMA-VLBI data show that the current Julia implementation provides a comparable performance to a highly optimized Fortran/C++ imaging code despite the fact that it is still mostly single-threaded and not yet fully optimized. Furthermore, the Julia implementation provides a significant improvement in the fidelity of the reconstructions for high dynamic range imaging.

This ALMA Study program has demonstrated that Julia is a promising framework for the further development of software packages and algorithms for ALMA imaging. All the packages developed under this program are publicly available in the software repository on GitHub, and can be installed with Julia's package manager in the standard ways. The software suites, which can natively handle multidimensional data products for ALMA imaging, enable the further development and implementation of new imaging algorithms. We will continue to develop software packages under the software suite for the broader ALMA community following the conclusion of this program.

ACKNOWLEDGMENTS

This study has been supported by the North American ALMA Cycle 8 Development Study Program of the National Radio Astronomy Observatory. We thank Paul Tiede, Lindy Blackburn, and Chi-Kwan Chan for many useful discussions regarding Julia and other modern scientific computing platforms. This paper makes use of the following ALMA data: ADS/JAO.ALMA2012.1.00631.S. ALMA is a partnership of ESO (representing its member states), NSF (USA), and NINS (Japan), together with NRC (Canada), MOST, and ASIAA (Taiwan), and KASI (Republic of Korea), in cooperation with the Republic of Chile. The Joint ALMA Observatory is operated by ESO, AUI/NRAO, and NAOJ. The National Radio Astronomy Observatory is a facility of the National Science Foundation operated under cooperative agreement by Associated Universities, Inc.

REFERENCES

Akiyama, K., & Matthews, L. D. 2019, ngVLA Memo, No. 66, arXiv:1910.00013

Akiyama, K., Kuramochi, K., Ikeda, S., et al. 2017a, ApJ, 838, 1

Akiyama, K., Ikeda, S., Pleau, M., et al. 2017b, AJ, 153, 159

Barnett, A. H., Magland, J., & af Klinteberg, L. 2019, SIAM Journal on Scientific Computing, 41, C479

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. 2017, SIAM Review, 59, 65

Bouman, K. L., Johnson, M. D., Zoran, D., et al. 2016, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 913

Broderick, A. E., Gold, R., Karami, M., et al. 2020, The Astrophysical Journal, 897, 139

Carpenter, J., Iono, D., Kemper, F., & Wootten, A. 2020, arXiv e-prints, arXiv:2001.11076

Chael, A., Issaoun, S., Pesce, D. W., et al. 2022, arXiv e-prints, arXiv:2210.12226

Chael, A., Narayan, R., & Johnson, M. D. 2019, MNRAS, 486, 2873

Chael, A. A., Johnson, M. D., Bouman, K. L., et al. 2018, ApJ, 857, 23

Chael, A. A., Johnson, M. D., Narayan, R., et al. 2016, ApJ, 829, 11

Cornwell, T. J. 2008, IEEE Journal of Selected Topics in Signal Processing, 2, 793

Event Horizon Telescope Collaboration, Akiyama, K., Alberdi, A., et al. 2019a, ApJL, 875, L2

—. 2019b, ApJL, 875, L4

Högbom, J. A. 1974, A&AS, 15, 417

Johnson, M. D., Bouman, K. L., Blackburn, L., et al. 2017, ApJ, 850, 172

Keiner, J., Kunis, S., & Potts, D. 2009, ACM Transactions on Mathematical Software (TOMS), 36, 1

Knopp, T., Boberg, M., & Grosser, M. 2022, arXiv e-prints, arXiv:2208.00049

Kuramochi, K., Akiyama, K., Ikeda, S., et al. 2018, ApJ, 858, 56

Matthews, L. D., Reid, M. J., Menten, K. M., & Akiyama, K. 2018, AJ, 156, 15
Tiede, P. 2022, Journal of Open Source Software, 7, 4457
Yamaguchi, M., Akiyama, K., Tsukagoshi, T., et al. 2020, ApJ, 895, 84