Development of auto-multithresh: an automated masking algorithm for deconvolution in CASA

Takahiro Tsutsumi,[1] Amanda Kepley,[2] IIsang Yoon,[2] and Urvashi Rau[1]

[1]*National Radio Astronomy Observatory, Socorro, NM, USA;*
ttsutsum@nrao.edu

[2]*National Radio Astronomy Observatory,Charlottesville, VA, USA*

Abstract.
A general purpose automated masking algorithm for deconvolution was developed in order to support automated data processing in ever-increasing data volumes of the current and future radio interferometers as described by Kepley et al in this meeting (Kepley et al. (2019)). In this presentation, we describe some technical details of the implementation of the automated masking algorithm named, "auto-multithresh", which was integrated into the refactored imaging task (*tclean*) in CASA. We also discuss our approach that we took for the development, which loosely follows the iterative model, so that the implementation is refined progressively for its functionality and performance based on testing and updated requirements throughout prototyping in Python to the final production in C++.

1. Auto-multithresh algorithm

A basic concept is to mimic interactive masking done experienced astronomers during CLEAN. A user can control the parameter setting through the quantities such as *rms* noise, sidelobe level, and synthesized beam size. Figure 1 shows some of the key features of the process and more detailed description can be found in the CASA Docs[1].

1.1. The Key features

The following are the key features of the algorithm.

- Iterative (run at the beginning of minor cycle)
- Threshold based mask created using a current residual image
- "Prune" : mechanism to remove unrealistic (or noise like) mask regions – regions smaller than user-specifiable parameter in fractions of synthesized beam
- "Grow": grow the threshold based mask to include low surface brightness regions using a binary dilation algorithm

[1]e.g. for CASA 5.4.0 documentation: https://casa.nrao.edu/casadocs/casa-5.4.0/synthesis-imaging/masks-for-deconvolution

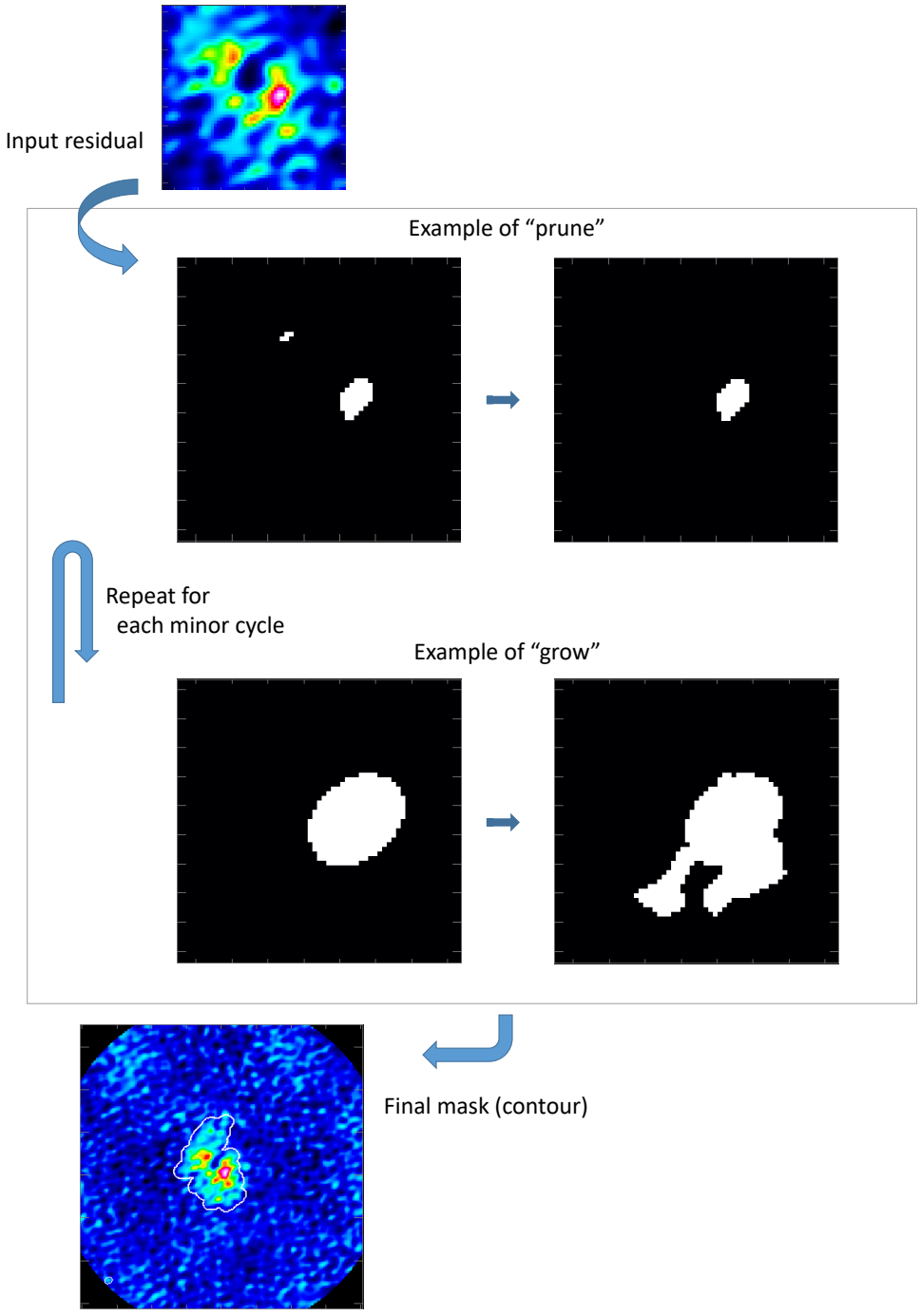

Figure 1. Auto-multithresh process (showing only some of key features)

- Handle negative (absorption) and positive (emission) features. It tracks the two features separately to avoid interaction.

- For cube imaging, allow to skip channels for no mask or no mask change from the previous iteration

2. Implementation Details

The prototype algorithm development was done in Python. The modular design of the refactored imaging code (C++ and Python) allows flexible implementation. The Figure 2 shows the code structure of the refactored imager. A wrapper Python class, PySynthesisImager, built on the top of the collection of the synthesis imaging Python tools. The tclean CASA task, which provide all imaging functionality, is built on top of PySynthesisImager (see Figure 2). Since each of the tools has one-to-one mapping of C++ classes and methods, prototyping by Python scripts can be easily accomplished using PySynthesisImager.

The final implementation of the algorithm was done in C++. The auto-multithresh algorithm is implentented in general mask handler class inside the refactor imaging code. It is launched from deconvolver to be in sync with its iteration control.

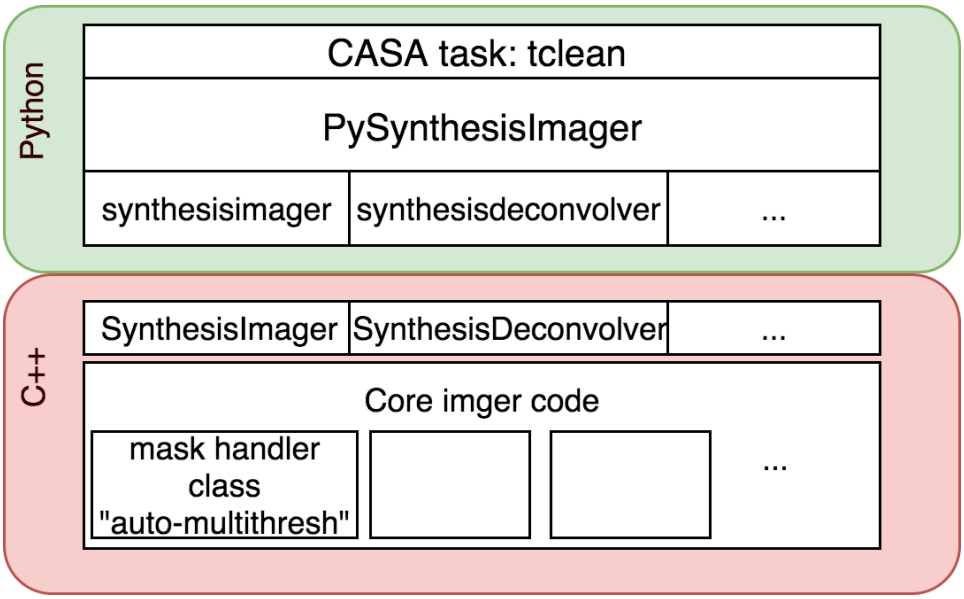

Figure 2. Implementation of auto-multithresh in CASA.

3. Development Process

The main driver of this development was coming from the ALMA pipeline. There was a research aspect to explore and refine an algorithm that work for the real ALMA data while meeting various time constraints including the CASA release schedules. It was necessary to adopt a development process slightly different from the standard CASA development, which generally completes within a single CASA development cycle. To do this we had a team of a dedicated developer for implementation and a scientist who led in design and verification as well as other testers for additional

scientific verifications. The process of this development follows (loosely) the iterative model.

1. Define requirements
2. the initial prototype development
3. Initial implementation
4. Verification/Validation Testing (performance, scientific correctness)
5. Amend or add to the requirements, if necessary
6. Implementation of additional features, mitigation to performance issues
7. repeat 4 -6

The adopted process generally worked well to deliver of the necessary functionality on time with flexibility of adding new features or making corrections in next iterations. However, one of the disadvantages was that the significant dedicated time by the key members for both code development and verification/validation testing was required. As for future projects of this nature, the observatory is making an effort to plan to separate resources for production from R&D efforts.

4. Current Status

The algorithm is availablesince CASA 5.0 release in *tclean* CASA task and various improvements were made ever since. It has been used in the production ALMA pipeline for Cycle 5 and beyond. While the original motivation was to be able use in ALMA imaging, it has been shown that the algorithm works on the data from other telescopes such as JVLA and ATCA

5. Future Development

For CASA 5.5 release, we plan to complete a bulk of the development, with introduction of a new noise estimate will be implemented to improve masking of absorption and extended emission. As a future research, we plan to explore to improve code efficiency by moving a part of the algorithm deeper inside the deconvolution algorithms

References

Kepley, A., Tsutsumi, T., Brogan, C., Indebetouw, R., Yoon, I., & Mason, B. 2019, in ADASS XXVIII, edited by TBD (San Francisco: ASP), vol. TBD of ASP Conf. Ser., TBD