Efficient Adaptive-Scale CLEAN Deconvolution in CASA for Radio Interferometric Images

M. Hsieh & S. Bhatnagar

January, 2021

Executive Summary

Scale sensitive solvers are widely used for accurate reconstruction of extended emission in radio astronomy. The Adaptive Scale Pixel decomposition (Asp) algorithm models the sky brightness by adaptively determining the optimal scales. It thus gives a significantly better imaging performance, but at a cost of significantly increased computational time. In this report, we described an improved Asp algorithm which achieves 3x-20x speed up in computational time comparing to the original Asp-Clean algorithm. It is also combined with the scale-insensitive Högbom CLEAN algorithm to achieve even better computational efficiency for both compact and diffuse emission. We implemented the algorithm in CASA and applied it to data sets from EVLA and ALMA telescopes. We show that this algorithm has performed better than the wide used MS-Clean algorithm. It has also achieved imaging performance without the need for hand-tuning of scale sizes or an expensive automasking algorithm, typically used in pipeline processing (like the current ALMA imaging pipeline). In summary, we have a working implementation of the basic (narrow-band) Asp-Clean integrated with CASA's imaging framework, and is available by specifying `deconvolver='Asp'` at the task-level tclean(). We plan to research new techniques to extend the algorithm to handle wide-band emission as well as utilizing GPU to further improve computational efficiency.

Contents

1 Background 3

2 Improved Asp-Clean Algorithm 4
2.1 Changes 8
2.1.1 Update model and residual image using the latest aspen 8
2.1.2 Normalization methods for initial guess 8
2.1.3 Fused deconvolution 10
2.1.4 Objective function and derivatives for optimization 10
2.1.5 Optimization third party libraries 11

3 Asp-Clean Imaging Results 13
3.1 Simulated Point Source 13
3.2 M31 14
3.3 G055.7+3.4 14
3.4 ALMA M100_12m_7m 16
3.5 CygA 16
3.6 ngVLA Imaging 19
3.7 EVLA Simulation 19
3.8 Performance Comparison with Other CLEAN algorithms 20

4 Ongoing and Future Work 24

Chapter 1

Background

The most widely used deconvolution algorithms in the radio synthesis imaging field are CLEAN algorithms. Scale-insensitive CLEAN algorithms, originally developed by Högbom (1974), decompose the true sky image as a collection of point sources or scaled delta functions. However, it needs a huge number of scaled delta functions to approximate diffuse emission and complex images, and it lacks a mechanism to introduce the dependence among the neighboring points. Thus the residuals of extended sources often include some significantly correlated structures. Scale-sensitive CLEAN algorithms, such as the Multi-Scale CLEAN (MS-Clean) algorithm, extends the CLEAN algorithms by assuming that astronomical images consist of a finite set of components (called "scales") with varying sizes (Cornwell (2008)). It is good for both point-like emission and diffuse emission. However, it is restricted to predefined set of scale sizes. Asp-Clean algorithm (Bhatnagar & Cornwell (2004)) uses an optimization technique to overcome this problem by keeping the size of the components variable. This work improved the Asp-Clean algorithm and implemented it in CASA, and we plan to extend it to handle wide-band emission.

Chapter 2

Improved Asp-Clean Algorithm

The original Asp-clean algorithm is described in Bhatnagar & Cornwell (2004). Unlike MS-Clean, Asp-Clean does not need a user-provided list of scales but dynamically determines optimal scales. To accomplish this:

1. It first defines a set of initial scales. This is done by fitting a 2D Gaussian whose width is W to the PSF (see Figure 2.1), and the initial scales are defined as 0, W, $2W$, $4W$, and $8W$.
2. In each iteration, smooth the residual image by a Gaussian beam at 0, W, $2W$, $4W$, and $8W$.
   (a) Search for the global peak (F) among these smoothed residual images, and add the "Aspen" (i.e. Gaussian beam with width = optimal scale (e.g. $4W$), amplitude = F, centered at the location of the peak, (x, y), to the "active set".
   (b) Optimize the Aspen(s) in the "active set" by minimizing the objective function, $I_R - activeset * PSF$, where I_R is the residual image.
   (c) Compute the mode image and update the residual image.
   (d) Go to Step 2 unless the termination criteria is met or the residuals are noise-like.

Figures 2.2, 2.3, 2.4 briefly illustrate how the algorithm works.

The original Asp-Clean algorithm developed by Bhatnagar & Cornwell (2004) keeps two sets of Aspens, active set and the permanent list. Active set always keeps the current optimal scale(s), while the permanent list keeps the previous and the current optimal scales. In the Step 2 (c) above, it always evaluate all Aspens in the permanent list onto an empty model image, and always subtracts that full model image convolved with the PSF from the original dirty image. That is,

$$Model = Aspen1 + Aspen2 + \cdots + AspenN$$
$$Residual = OriginalDirtyImage - Model * PSF$$

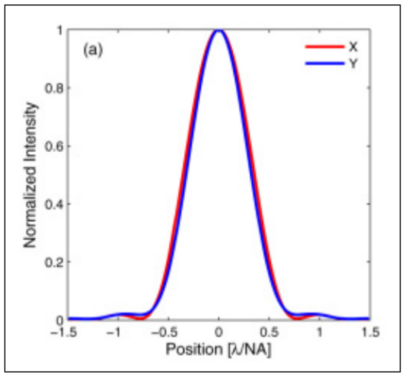


Figure 2.1: Fitting a Gaussian to PSF to get the PSF width

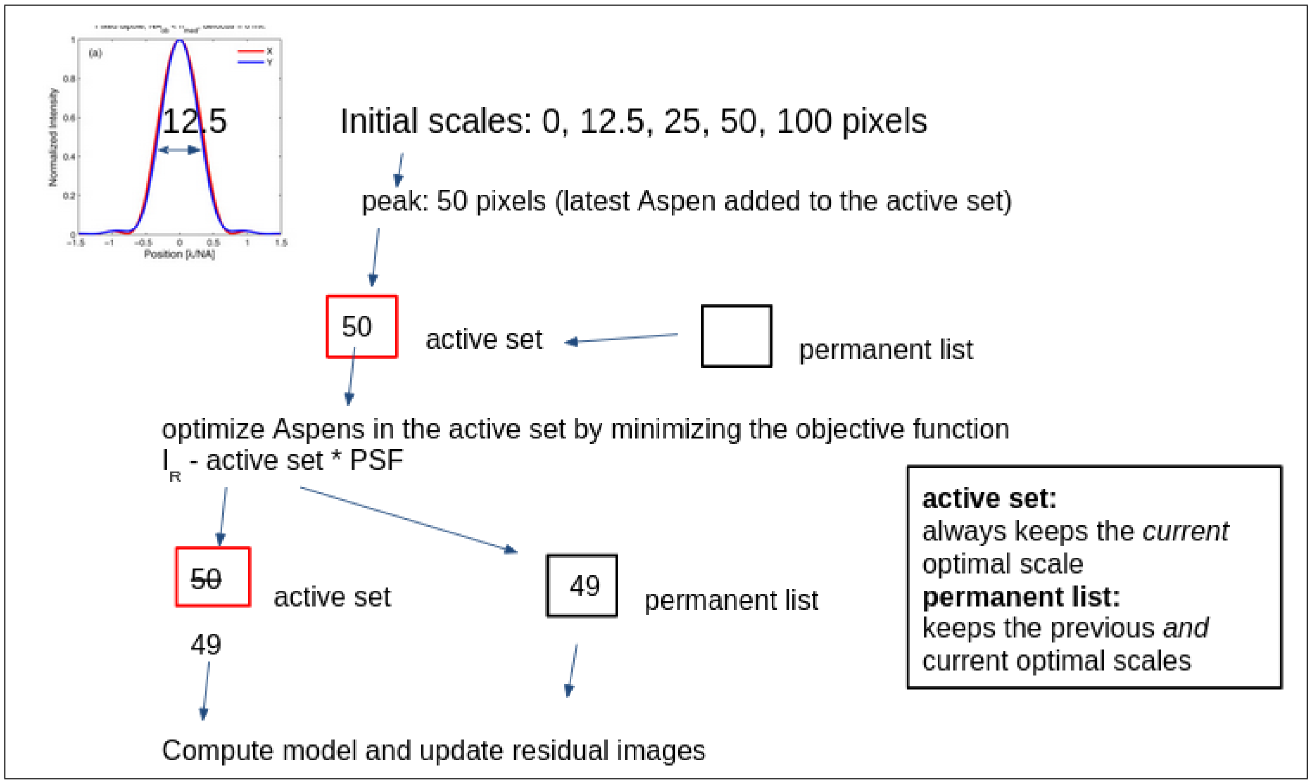
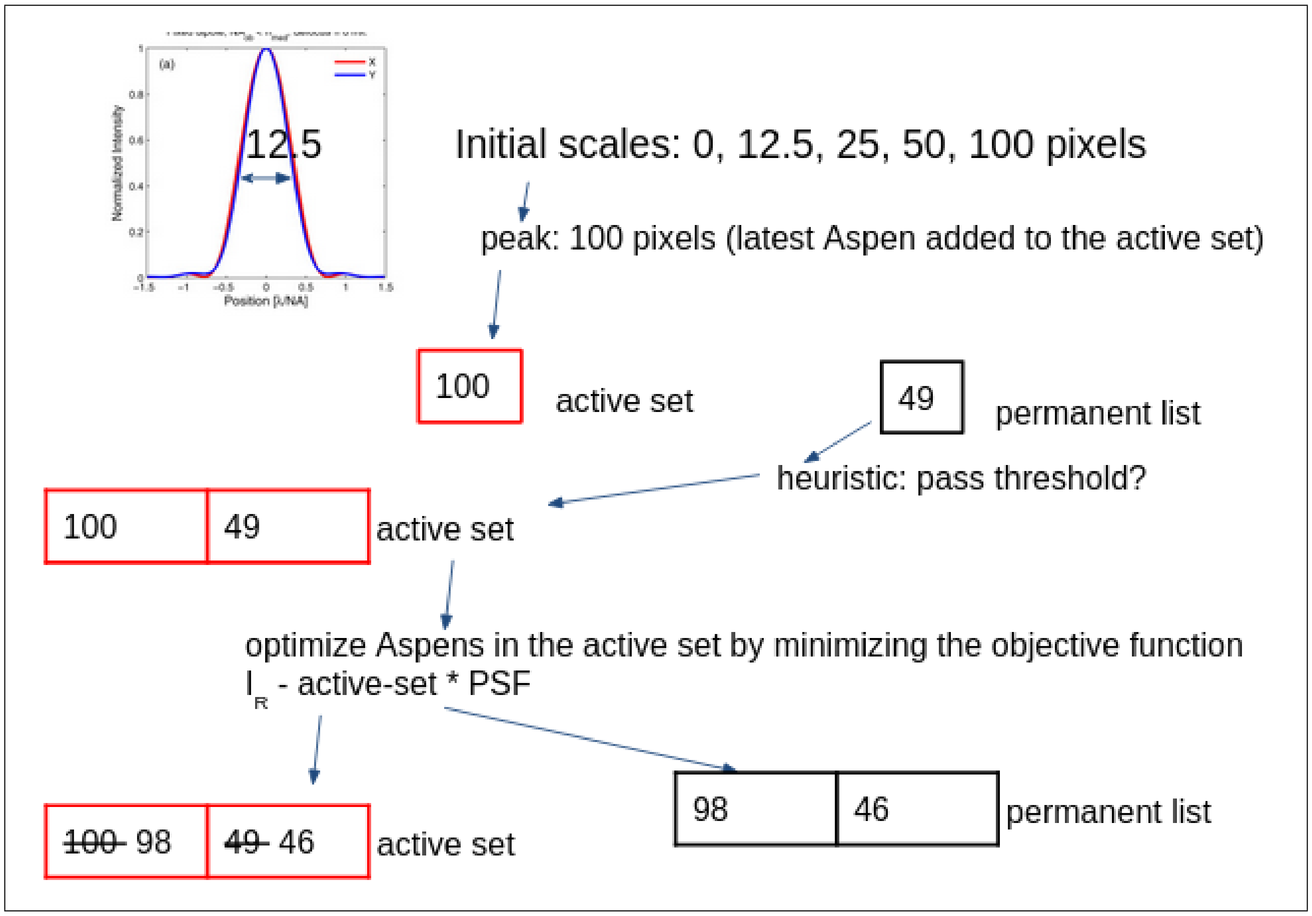

12.5
Normalized Intensity
Position [λ/NA]
Initial scales: 0, 12.5, 25, 50, 100 pixels
peak: 50 pixels (latest Aspen added to the active set)
50
active set
permanent list
optimize Aspens in the active set by minimizing the objective function
I_R - active set * PSF
~~50~~
49
active set
49
permanent list
active set:
always keeps the current optimal scale
permanent list:
keeps the previous and current optimal scales
Compute model and update residual images

Figure 2.2: Asp-Clean algorithm illustration (first iteration). In this example, the PSF width, W, is 12.5 pixels. The initial scales are 0, 12.5, 25, 50, and 100 pixels. In Step 2 of the algorithm, the residual image is smoothed by these scales, and the global peak, F, is found at the residual smoothed by a Gaussian beam at 50 pixels at location (x, y). An Aspen (i.e. Gaussian beam with width = 50, amplitude = F, centered at the location, (x, y)), is added to the “active set”. The active set which only has one Aspen in this case is optimized and the scale size of the Aspen is optimized from 50 to 49 (the optimization on the amplitude is not shown here). The optimized Aspen is then added to the permanent list. The model and the residual images are updated accordingly.



12.5
Initial scales: 0, 12.5, 25, 50, 100 pixels
peak: 100 pixels (latest Aspen added to the active set)
100
active set
49
permanent list
heuristic: pass threshold?
100
49
active set
optimize Aspens in the active set by minimizing the objective function
I_R - active-set * PSF
~~100~~ 98
~~49~~ 46
active set
98
46
permanent list

Figure 2.3: Asp-Clean algorithm illustration (second iteration). Similar to Figure 2.2, in this iteration the global peak is found at the residual smoothed by a Gaussian beam at 100 pixels. An Aspen with scale size = 100 is added to the "active set". The Aspen in the permanent list (the optimized scale from the last iteration) is also added to the active set based on a heuristic approach described in Bhatnagar & Cornwell (2004). The active set now has two Aspens. The scale sizes of the two Aspens are optimized to 98 and 46. The permanent list is updated with these two optimized Aspens. The model and the residual images are updated accordingly.

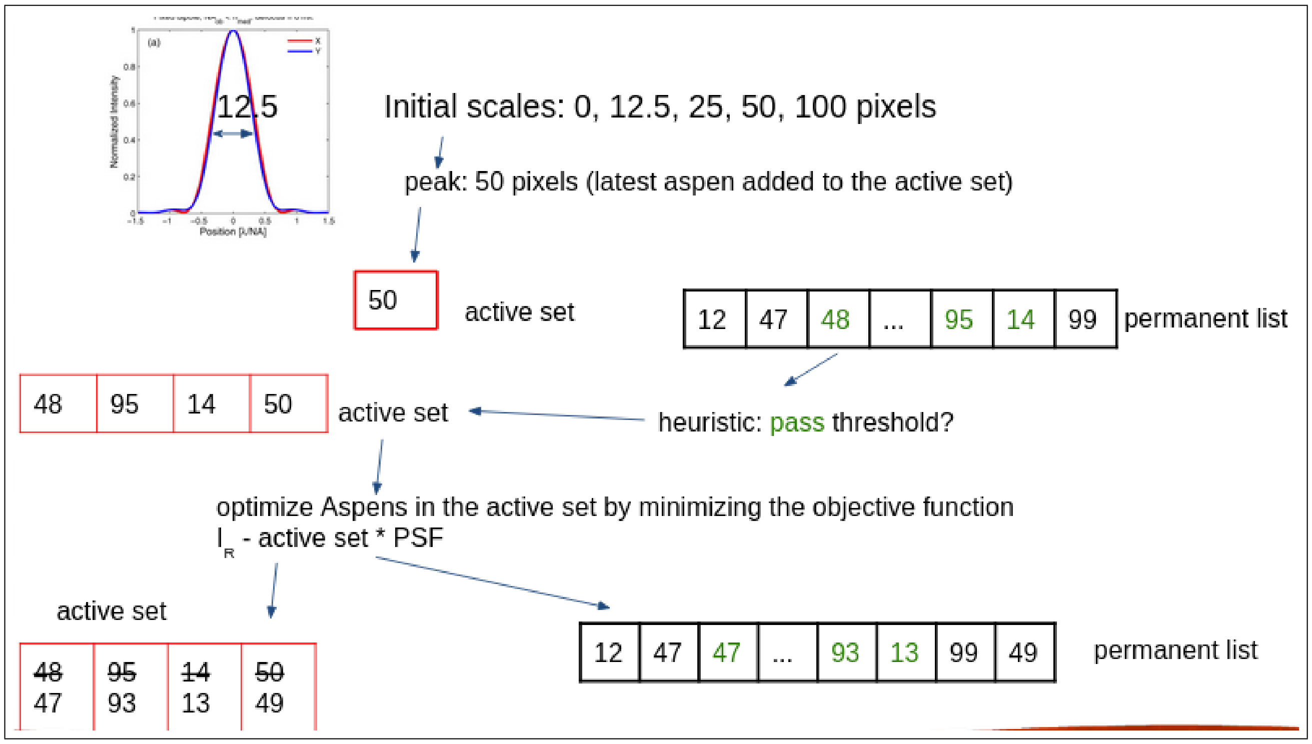


Figure 2.4: Asp-Clean algorithm illustration (nth iteration). Similarly, in this iteration the global peak is found at the residual smoothed by a Gaussian beam at 50 pixels. An Aspen with scale size = 50 is added to the “active set”. There are several Aspens in the permanent list and only the Aspens with scale size = 48, 95, and 14 are added to the active set based on the heuristic approach. The active set now has four Aspens. The scale sizes of the four Aspens are optimized. The permanent list is updated with these four optimized Aspens. The model and the residual images are updated accordingly.

2.1 Changes

The original Asp-Clean algorithm gives a significantly better imaging performance, but Step 2 (b) becomes inefficient for complex images where the total number of Aspens can be several hundreds. The Bhatnagar & Cornwell (2004) paper described a heuristic approach that removes Aspens which were significant to begin with but became insignificant in the later cycles to speed up the computation.

In this work, we improved the original approach described in Step 1 of Chapter 2 for defining the initial scale sizes to prevent WAsp from picking a large scale that has no constraints from the data (Section 2.1). We also developed a new approach that simplified the algorithm (see Section 2.1.1) and reduce the runtime further without degrading the imaging performance. Moreover, the original Asp-Clean implementation details are not described in the Bhatnagar & Cornwell (2004) paper (e.g. normalization method for selecting an initial scale). Fortunately, we have the code developed for that work and used that as a starting point to re-derive the objective function and its derivatives for Aspens optimization (Section 2.1.3) as well as implemented our own normalization methods (Section 2.1.2). Furthermore, we developed a fused deconvolution algorithm that combines the scale-insensitive Högbom CLEAN algorithm and the Asp-Clean to further improve the imaging performance and computational efficiency (Section 2.1.3).

Define Initial Scale Sizes

The original Asp-Clean defines the initial scale sizes to be 0, W, $2W$, $4W$, and $8W$, where $8W$ may be too large to have constraints from the data. Therefore, we provided a user-defined tclean parameter,$largestscale$, in CASA that allows to overwrite the default (i.e. $8W$). This prevents Asp-Clean from inadvertently fitting large scale negative sidelobes in extreme cases, like the jet dataset (Section 3.7).

When the $largestscale$ is set, the initial scale sizes are qauranteed to range from 0, W (i.e. the width of PSF), $2W$, up to the $largestscale$ size.

2.1.1 Update model and residual image using the latest aspen

We simplified the Step 2 (c) of the original Asp-Clean algorithm by only using the latest Aspen of each iterations to update the model and the residual images. That is,

$$Model+ = LatestAspen$$
$$Residual- = LatestAspen * PSF$$

Therefore, every iteration is like Figure 2.2 where there is only one Aspen in the active set for optimization and there is no need of a permanent list. The Asp-Clean algorithm is to discover a basis set adopted to the structure in the image, and the various heuristics are there to handle a covariance matrix that is not diagonal at all. This change keeps only the latest Aspen in the active set and is approximating the matrix as a diagonal, which is also what many optimization algorithms do and then iterate. Therefore, it fits well in the larger theory as well. Using this approach, the runtime is reduced by 3x-20x comparing to the original approach and their imaging performance are similar.

2.1.2 Normalization methods for initial guess

Normalization in the Step 2 (a) is critical on providing a good initial guess of the scale size and the amplitude of an Aspen for optimization. Normalization here does not refer to the normalization of Gaussian components to have unit area. Normalization method should be designed to avoid consequent 0 scales at the beginning, and also the normalized optimal strength (i.e. initial guess of the amplitude of an Aspen) cannot be too large. Otherwise, optimization algorithms will return very large scale size. We developed two normalization methods (Sections 2.1.2 and 2.1.2), compared their performance (Section 2.1.2) and provided users a way to choose either one of the methods (Section 2.1.2).

Normalization Method 1

The steps below describes how Normalization Method 1 is used in the Asp-Clean.

1. Convolved the residual image with the five initial scales, resulting $ResConvInitScale[0-4]$. The initial scales are [0, $PsfWidth$, 2 * $PsfWidth$, 4 * $PsfWidth$, 8 * $PsfWidth$] as mentioned earlier.

2. Find the global peak among $ResConvInitScale[0-4]$. The global peak is denoted, $itsStrengthOptimum$.
3. Normalize $itsStrengthOptimum$ by Normalization Method 1, and this becomes the initial guess of the amplitude of an Aspen for optimization.

The Normalization Method 1 was developed based on the code of the original work (Bhatnagar & Cornwell (2004)). It was modified to give the best results with our simplified approach to update the residual image. The mathematical details are described below.

1. Initial scale is modeled by a 2D Gaussian component:

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} \tag{2.1}$$

However, this expression does not have unit area. To have a unit area, the 2D Gaussian component should have the following expression:

$$\frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} \tag{2.2}$$

So we can actually see Eq. 2.1 as normalizing Eq. 2.2 by $\frac{1}{\sigma\sqrt{2\pi}}$. Then, the residual image, $Residual$, is convolved with the five initial scales as followed.

$$ResConvInitScale[i] = Residual * \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma_i^2}} \tag{2.3}$$

for each of the five scales, σ_i, except the 0 scale.

2. Find the global peak among $ResConvInitScale[0-4]$.

$$itsStrengthOptimum = \text{global peak}(ResConvInitScale[i]) \tag{2.4}$$

3. The initial guess of the amplitude of an Aspen is calculated by

$$InitialGuessAmplitude = \frac{itsStrengthOptimum}{normalization} \tag{2.5}$$

, where

$$normalization = \sqrt{\frac{2\pi}{d}} \tag{2.6}$$

, $d = \sqrt{\frac{1}{PsfWidth^2} + \frac{1}{\sigma_{opt}^2}}$ and σ_{opt} is the scale that gives the global peak in Step 2.

Normalization Method 2

The motivation to develop another normalization method is because Normalization Method 1 rarely picks 0 scale. This ends up doing little residual update with large components whose amplitude is very small at later iterations when larger scale emissions are already removed. The Normalization Method 2 allows more 0 scales at random time. The approach was inspired by what MS-Clean does for finding the global peak and its main differences from Normalization Method 1 are shown below.

1. This is the same as the Step 1 of Normalization Method 1 (Section 2.1.2).
2. Normalize $ResConvInitScale[0-4]$ first and then find the global peak among them. That is,

$$itsStrengthOptimum = \text{global peak of } \left(\frac{ResConvInitScale[i]}{\sqrt{2\pi\sigma_i}}\right) \tag{2.7}$$

3. The initial guess of the amplitude of an Aspen is calculated by

$$InitialGuessAmplitude = \frac{itsStrengthOptimum\sqrt{2\pi\sigma_{opt}}}{normalization} \tag{2.8}$$

, where $normalization$ is defined in Eq. 2.6 and σ_{opt} is the scale that gives the global peak in Step 2.

Normalization method option in CASA

We provide an environment variable, ASP_NORM, for users to choose either one of the normalization methods. To choose Normalization Method 1 (this is the default), set `ASP_NORM=1`. To choose Normalization Method 2, set `ASP_NORM=2`.

Evaluation

We ran Asp-Clean with Normalization Method 2 on the M31 and the G055.7+3.4 datasets (Section 3.2 and 3.3, and both have very good results (residual images are much more noise-like than MS-Clean). However, it performs poorly on the simulated point source dataset (Section 3.1) because it only can picks 0 scales. On the other hand, although Normalization Method 1 rarely picks 0 scales, we found out as long as it is used along with Högbom CLEAN in the fused approach described in Section 2.1.3, it give good results on all datasets presented in the memo. We therefore recommends to always first try Normalization Method 1 and adjust the threshold, `fusedthreshold`, until a good balance of imaging performance and computational efficiency is achieved.

2.1.3 Fused deconvolution

Due to the reason that Normalization Method 1 rarely picks 0 scale as well as to further reduce the Asp-Clean runtime, Asp-Clean can automatically be "switched to Högbom CLEAN" when either of the following two criteria is met. Note that in this memo, by "switched to Högbom CLEAN" we mean Asp-Clean would only use point-source flux components for a number of iterations until it is "switched back to Asp-Clean".

1. Peak residual is smaller than a user-provided threshold, `fusedthreshold`. The tip to set this threshold is to run tclean with the Asp-Clean deconvolver in the interactive mode without setting `fusedthreshold`. When larger scale emissions are removed from the residual image or when the peak residual has rarely changed for many iterations, the peak residual at that time would be the ideal value for `fusedthreshold`. Then, re-run tclean with `fusedthreshold` set.
2. In ten consecutive iterations, when more than five iterations picks 0 scales, Asp-Clean is switched to Högbom CLEAN (See Zhang (2018)).

When Högbom CLEAN is "triggered" (i.e. Asp-Clean would only use point-source flux components), it is ran for the following number of iterations and then switched back to Asp-Clean.

$$NumHogbomIter = \text{ceil}(50 + 2(e^{0.05tn} - 1)); \tag{2.9}$$

, where $NumHogbomIter$ is the number of iterations to run Högbom CLEAN when Högbom CLEAN has been triggered tn^{th} times. As described in Zhang (2018), the specific form of this function is not important. Therefore, in the later development, we simplified the equation and let "Högbom CLEAN run" (i.e. Asp-Clean would only use point-source flux components) for 51 iterations. If it is approaching convergence, Högbom CLEAN would run for longer iterations, 510 iterations, instead (see also Section 4). The fused deconvolution saves computational time without degrading imaging performance.

2.1.4 Objective function and derivatives for optimization

The objective function that we want to minimize for optimization is

$$\chi^2 = ||I_R - a \cdot (PSF * Aspen)||^2 \tag{2.10}$$

, where $Aspen$ is defined in Eq. 2.1, I_R is the residual image and a is the amplitude of the Aspen.
The partial derivatives of χ^2 with respect to amplitude and scale are:

$$\frac{\partial \chi^2}{\partial a} = -2(I_R - a \cdot (PSF * Aspen)) \cdot (PSF * Aspen) \tag{2.11}$$

$$\frac{\partial \chi^2}{\partial \sigma} = -2(I_R - a \cdot (PSF * Aspen)) \cdot (a \cdot (PSF * \frac{Aspen}{\sigma}) \cdot (\frac{(x-x_0)^2 + (y-y_0)^2}{\sigma^2} - 1)) \tag{2.12}$$

It is worth noting that the matrix multiplication operator here, $\cdot$, should be element by element.

2.1.5 Optimization third party libraries

The original Asp-Clean implementation used GNU Scientific Library (GSL) for optimization in 2004. Since then, many third party libraries have been developed for better computing performance. All of them have different API but the basic requirement is to pass an objective function for optimization through the API. The derivatives of the objective function may no be required since some libraries can do the optimization without derivatives. We researched and first integrated LBFGS++ that is widely used in the academia. However, we encountered several challenges (Section 2.1.5) and tried three other libraries as well. All of these libraries are listed below. We ended up using GSL, same as the original Asp-Clean work, to ensure our code case is comparable with it. Our optimization code is verified by fitting an Aspen to a simple dirty image (Gaussian component $*$ real PSF).

1. LBFGS++ (Qiu (2015)). It is a header-only C++ library that implements the Limited-memory BFGS algorithm (L-BFGS) for unconstrained minimization problems, and a modified version of the L-BFGS-B algorithm for box-constrained ones. Its API is easy to work with. It requires the Eigen library which is already part of the CASA build.

2. CppNumericalSolvers (Wieschollek (2016)). It is a lightweight C++17 library of numerical optimization methods for nonlinear functions. Its API is similar to LBFGS++, but returns very different results comparing to LBFGS++.

3. ALGLIB (Bochkanov (1999)). It is a cross-platform numerical analysis and data processing library and provides various optimization methods.

4. GSL (M. Galassi et al. (2009)). It is a numerical library for C and C++ programmers and provides wide range of mathematical routines, including optimization. The API is hard to work with and can easily cause segmentation fault if not using the API correctly. Debugging segmentation fault with `gdb` is not useful and requires special GSL API for debugging. We created a new class for passing Aspens and data between CASA and GSL. We found out GSL is more stable than the libraries above, and this is probably because GSL provides a restart function when an optimization step fails. GSL provides two methods for optimizing without derivatives, `gsl_multimin_fminimizer_nmsimplex2` and `gsl_multimin_fminimizer_nmsimplex2ran`. However, the former returns very large scale sizes. The latter gives better optimization result but its runtime is 3x longer than the optimization method with derivatives. Therefore, we ended up using GSL optimization with derivatives in our implementation.

After the first version of this memo was finished, we did a detailed performance analysis of Asp-Clean and identified the BFGS optimization as the performance bottleneck (Section 3.8). Therefore we switched the GSL with the ALGLIB, which improves the Asp-Clean runtime by at least 4x with similar imaging performance (Figure 3.17). All of the Asp-Clean results in this memo were done with the GSL library unless specified otherwise. It was later found out that Asp-Clean with ALGLIB can have variability across different build systems. This is because of the following reasons and is decided to be acceptable considering its runtime improvement over the GSL.

1. The BFGS works by approximating the Hessian and invokes matrix inversions. It is likely that Asp-Clean Hessian matrix has a large condition number that leads the algorithm astray. A large condition number implies that the matrix inversion is prone to numerical errors. This, coupled with round-off errors common in every floating point arithmetic system can lead to the results different results for same objective function. This is also seen in `scipy.optimze.fmin_bfgs`. One source of high condition number could easily be that the parameter Asp-Clean is optimizing is in the exponent (highly non-linear). If so, numerical stability might improve if transform the objective function for the purpose of optimization where the function is less non-linear. For example, in the original Asp-Clean implementation in 2004, the inverse of the scale in the exponent is used to try to reduce the condition number.

2. Floating point arithmetic is not universally defined and different parties can have different implementations. Different compilers and even the same compilers with different compiling options can change the order of computations, too. All these are likely to have big effect in ill-conditioned optimization.

3. The ALGLIB code is included in the CASA source tree and compiled together with the CASA code (that is what ALGLIB wants users to do). This makes the variability more noticeable when building with different versions of compilers. On the other hand, GSL is used by a linking with a pre-compiled shared library.

Challenges

Below lists a number of challenges we encountered when choosing and integrating the optimization libraries mentioned above.

1. Reliability and implementation details of third party BFGS optimization libraries are unclear. Therefore, it is hard to determine if an issue is due to a bug in the Asp-Clean code or the third party library without looking at the code of the library itself.
2. Each library provides several options of line search algorithms. Besides trying out each of them, it is hard to determine which works the best for the Asp-Clean without studying the line search algorithms themselves.
3. Each library has its own stopping criteria to prevent overfitting, but this is seldom documented in the manual. We needed to look at the code to understand what is done and most of the time needed to implemented our own stopping criteria.
4. Each library has different BFGS parameters for tuning. Setting the BFGS parameter values is not trivial either and requires many trials.
5. It is common to encounter BFGS fitting error like the following:
   - "NaN" is returned from the optimization library for the scale size of an Aspen. This is sometimes due to overfitting.
   - Overfitting. To fix this, some BFGS parameters need to be relaxed, like epsilon, tolerance for gradient, minimum and maximum step length, and number of iterations, etc).
   - Libraries return error message like, "exceeds maximum number of line search iterations" or "step length is too small". This is mainly due to the optimization got stuck in local minimum so the variables for optimizations are barely updated.
   - Negative scale size is returned. The original Asp-Clean implementation had this issue too and fixed it by always returning the absolute value of the scale size. We first tried the BFGS with bound constraints provided by the libraries (i.e. the scale size is bounded so it will not end up being negative). It works for simple example but for our case scale sizes can still go out of bound. We therefore ended up using the original approach to always return absolute value of scale sizes.
6. Computational time is long (majority of the Asp-Clean runtime spends on the optimization) so it makes debugging even harder.

Chapter 3

Asp-Clean Imaging Results

We applied the Asp-Clean to five data sets from EVLA and ALMA telescopes, and the results are shown in the following sections. In summary, this improved Asp-Clean algorithm has performed better than the wide used MS-Clean algorithm. It has also achieved imaging performance without the need for an expensive and complex automasking algorithm used in the ALMA imaging pipeline. The performance comparison between Asp-Clean, MS-Clean, and Högbom CLEAN is summarized in Section 3.8. A detailed performance analysis of Asp-Clean is shown in Section 3.8.

3.1 Simulated Point Source

The simulated point source is a simple test case for the Asp-Clean fused approach. Figure 3.1 compares the residual images of the simulated point source using the MS-Clean (left) and Asp-Clean (right). The MS-Clean was ran for 5000 iterations. The Asp-Clean was ran for about 2100 iterations and its residual is more noise-like than the MS-Clean. Figure 3.2 shows the restored images using the MS-Clean (left) and Asp-Clean (right). MS-Clean was run with four scales of sizes 0, 3, 5, and 10 pixels.

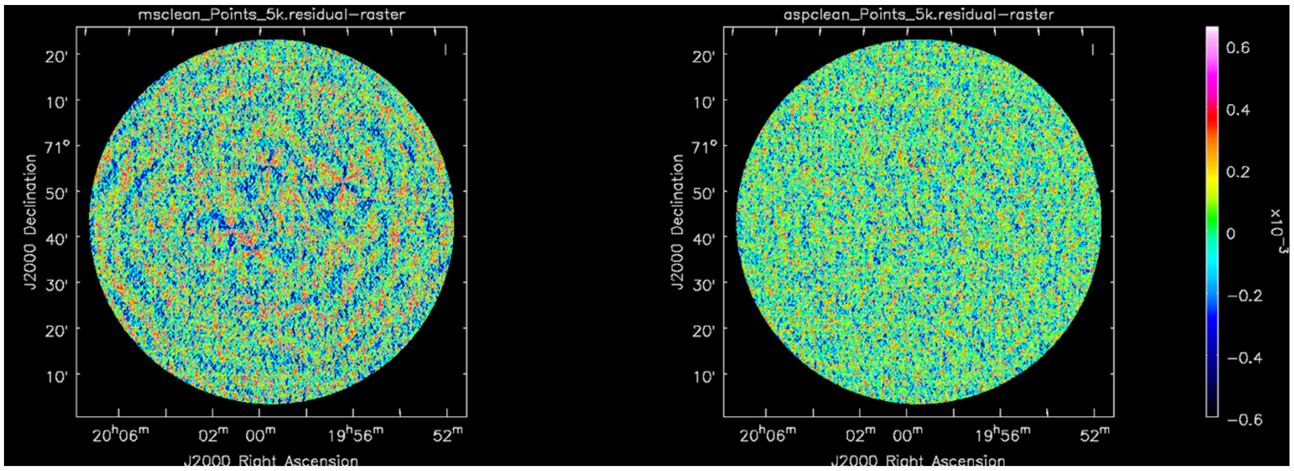


Figure 3.1: Residual Images of the simulated point source: MS-Clean vs. Asp-Clean

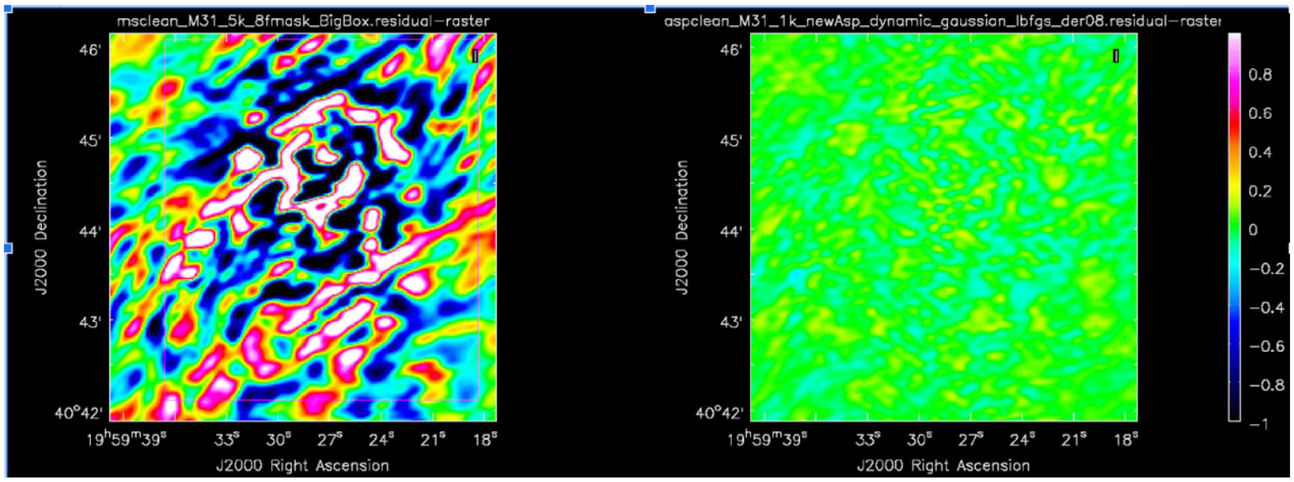

msclean_M31_5k_8fmask_BigBox.residual-raster
aspclean_M31_1k_newAsp_dynamic_gaussian_lbfgs_der08.residual-raster
J2000 Declination
J2000 Right Ascension

Figure 3.3: M31 Residual Images: MS-Clean vs. Asp-Clean

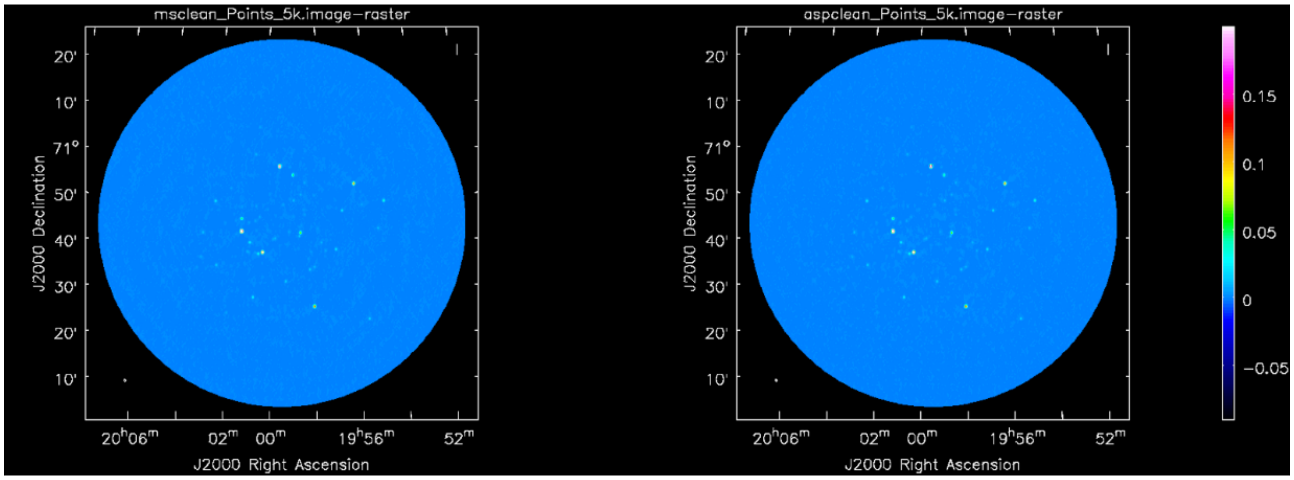

msclean_Points_5k.image-raster
aspclean_Points_5k.image-raster
J2000 Declination
J2000 Right Ascension

Figure 3.2: Restored Images of the simulated point source: MS-Clean vs. Asp-Clean

3.2 M31

The M31 data set is the simulated visibilities corresponding to a VLA observation. Figure 3.3 compares the M31 residual images using the MS-Clean (left) and Asp-Clean (right). The MS-Clean was ran for 5000 iterations. The Asp-Clean was ran for only 1000 iterations and its residual is already more noise-like than the MS-Clean whose residual is correlated with the large scale emission. Figure 3.4 shows the M31 restored images using the MS-Clean (left) and Asp-Clean (right). MS-Clean was run with five scales of sizes 0, 3, 5, 10 and 15 pixels.

3.3 G055.7+3.4

The G055.7+3.4 is a supernova remnant with a pulsar within it and is an extended source with many angular scales. Figure 3.5 compares the G055.7+3.4 residual images using the MS-Clean (left) and Asp-Clean (right). The MS-Clean was ran for 5000 iterations. MS-Clean was run with five scales of sizes 0, 6, 10, 30 and 60 pixels. The Asp-Clean was ran for about 1000 iterations and its residual is more noise-like than the MS-Clean. Figure 3.6 compares the G055.7+3.4 restored images using the MS-Clean (left) and Asp-Clean (right). The Asp-Clean restored image has less ripples and less negatives.

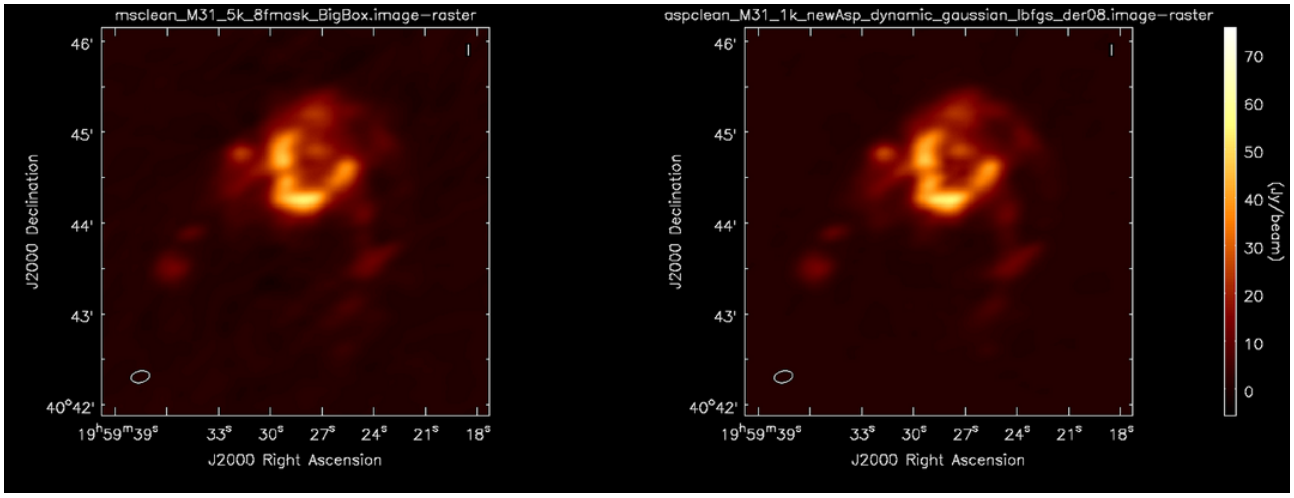


Figure 3.4: M31 Restored Images: MS-Clean vs. Asp-Clean

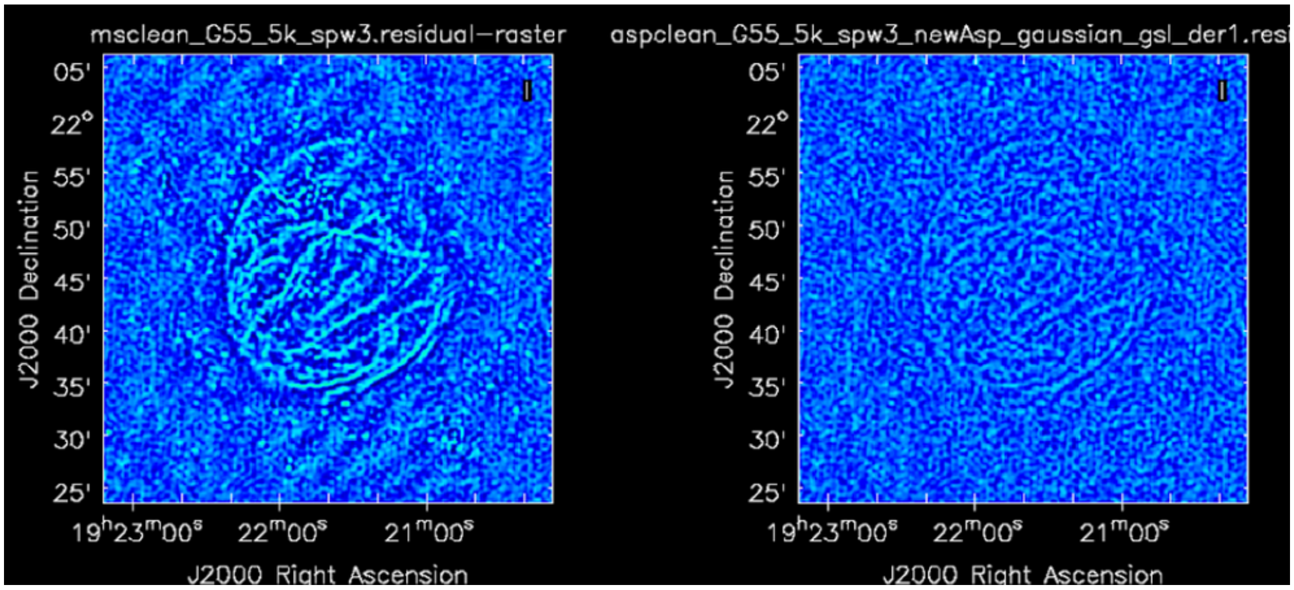


Figure 3.5: G055.7+3.4 Residual Images: MS-Clean vs. Asp-Clean

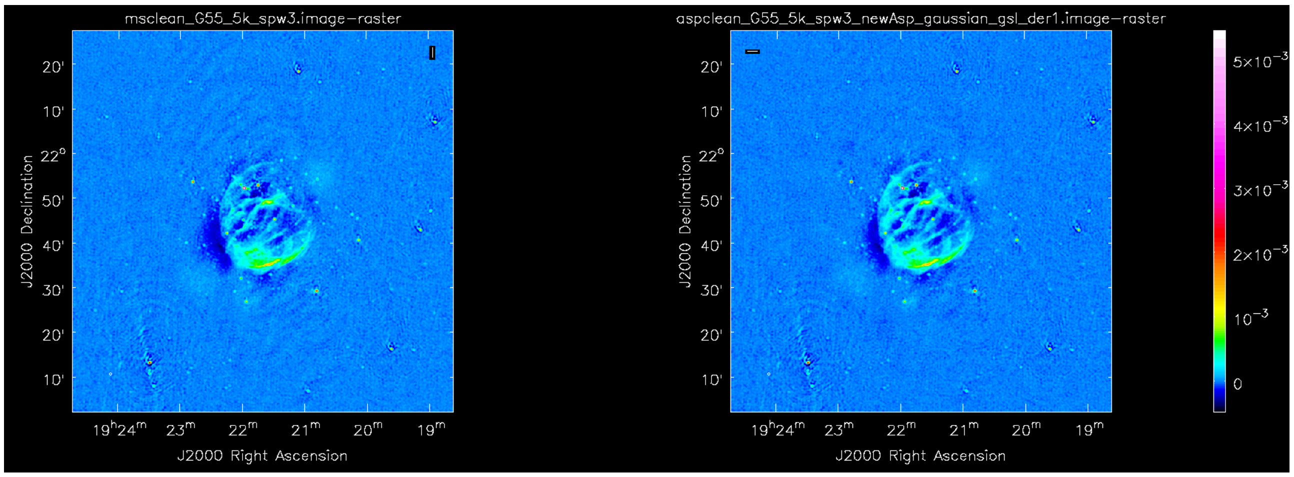


Figure 3.6: G055.7+3.4 Restored Images: MS-Clean vs. Asp-Clean

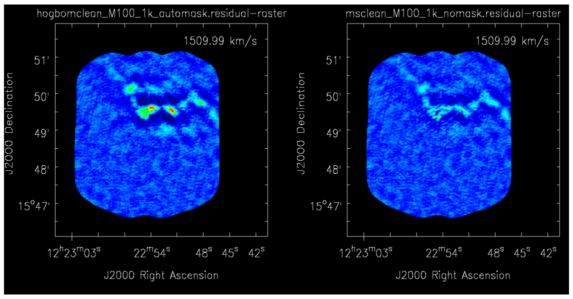


Figure 3.7: Residual Images of the M100_12m_7m Channel 22: Högbom CLEAN with automasking vs. MS-Clean

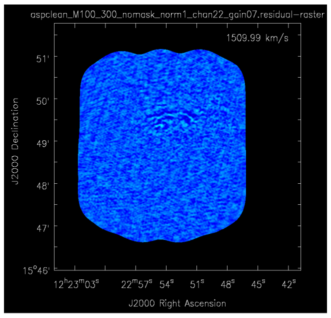


Figure 3.8: Residual Images of the M100_12m_7m Channel 22 with Asp-Clean

3.4 ALMA M100_12m_7m

Figures 3.7 and 3.8 compare the residual images of the M100_12m_7m Channel 22 using the Högbom CLEAN with automasking, MS-Clean without automasking and Asp-Clean without automasking. Figures 3.9 and 3.10 show the restored images of Channel 22. Both of the Högbom CLEAN and the MS-Clean were ran for 1000 iterations. The Asp-Clean was ran for 200 iterations. MS-Clean was run with three scales of sizes 0, 5, and 15 pixels. Figure 3.11 shows the restored images of Channel 24. Figure 3.12 shows the restored images of Channel 38. The results show that Asp-Clean performs better because it does not require automasking and the restored images have little negative bowls. It does not require hand-tuning of scale sizes like MS-Clean, either.

3.5 CygA

Figure 3.13 compares the residual images of the CygA VLA dataset at frequency 2.052GHz using the Högbom CLEAN with automasking, MS-Clean without automasking and Asp-Clean without automasking. MS-Clean was run with five scales of sizes 0, 10, 20, 30 and 60 pixels. Both of the Högbom CLEAN and MS-Clean were ran for 5000 iterations. The Asp-Clean was ran for about 4000 iterations and its residual is more noise-like than the MS-Clean. Figure 3.13 show the restored images.

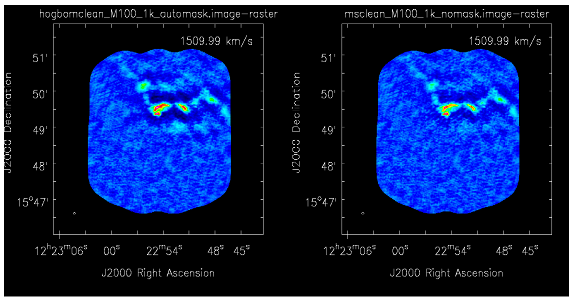


Figure 3.9: Restored Images of the M100_12m_7m Channel 22: Högbom CLEAN with automasking vs. MS-Clean

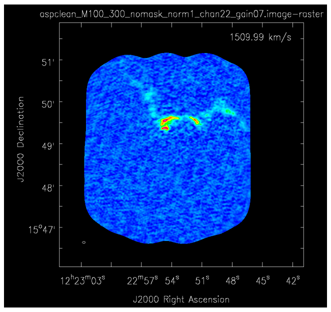


Figure 3.10: Restored Images of the M100_12m_7m Channel 22 with Asp-Clean

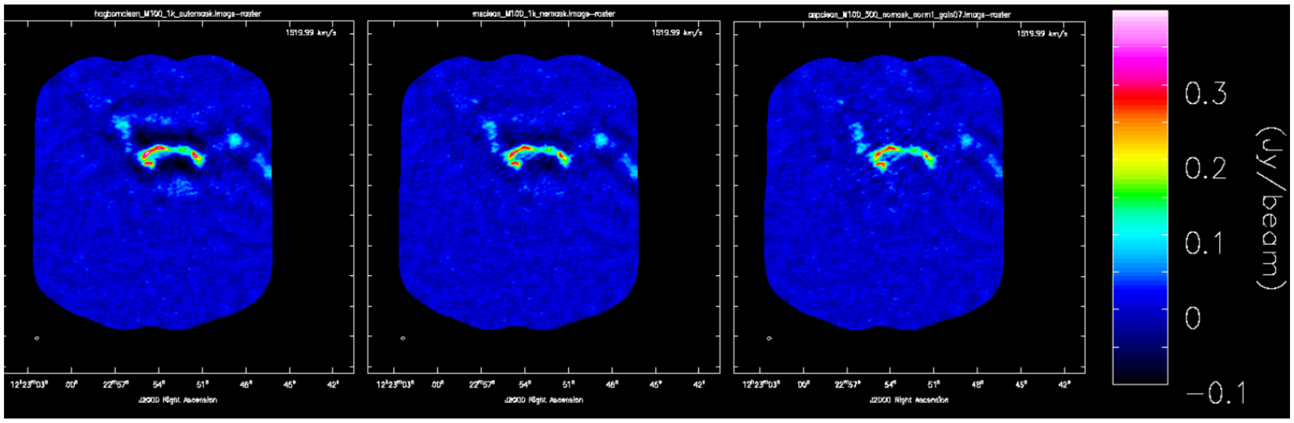


Figure 3.11: Restored Images of the M100_12m_7m Channel 24: Högbom CLEAN with automasking vs. MS-Clean vs. Asp-Clean

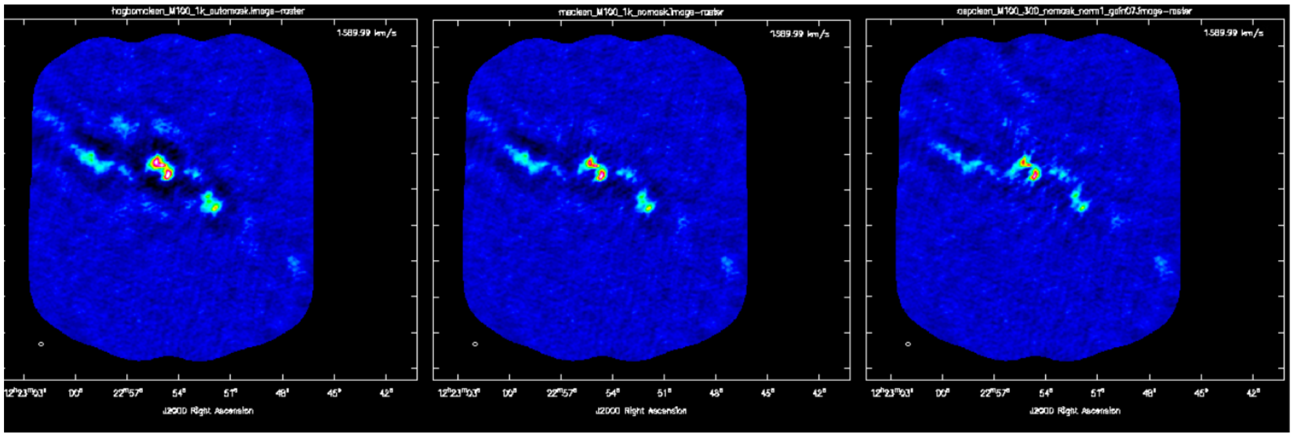


Figure 3.12: Restored Images of the M100_12m_7m Channel 38: Högbom CLEAN with automasking vs. MS-Clean vs. Asp-Clean

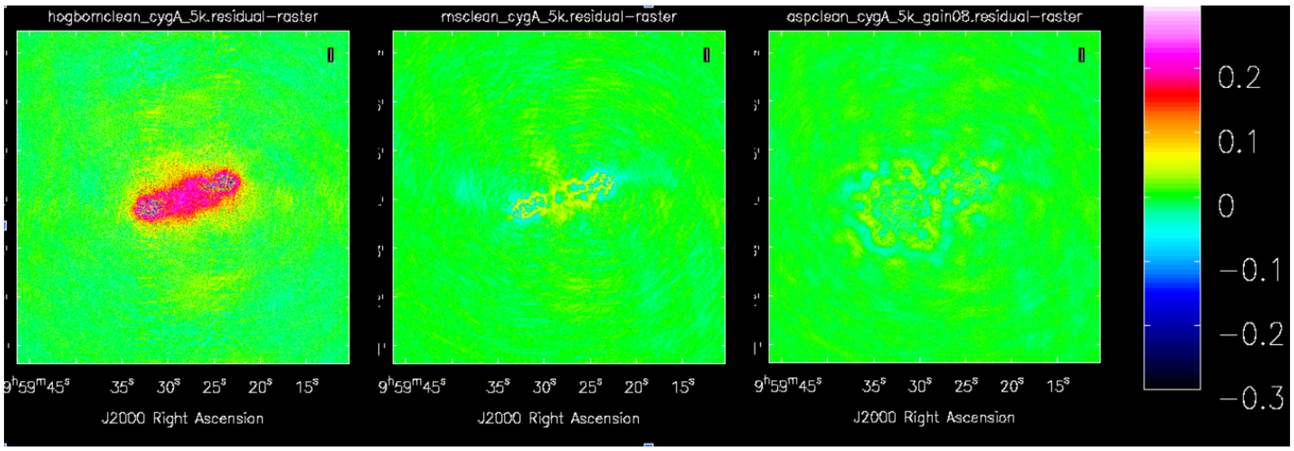


Figure 3.13: Residual Images of the CygA VLA dataset: Högbom CLEAN with automasking vs. MS-Clean vs. Asp-Clean

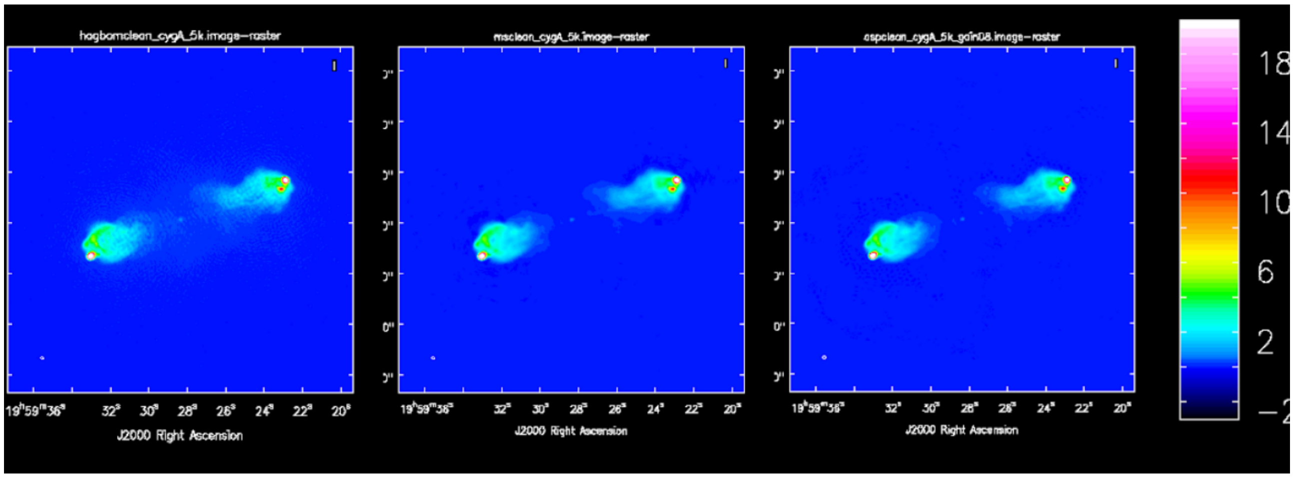


Figure 3.14: Restored Images of the CygA VLA dataset: Högbom CLEAN with automasking vs. MS-Clean vs. Asp-Clean

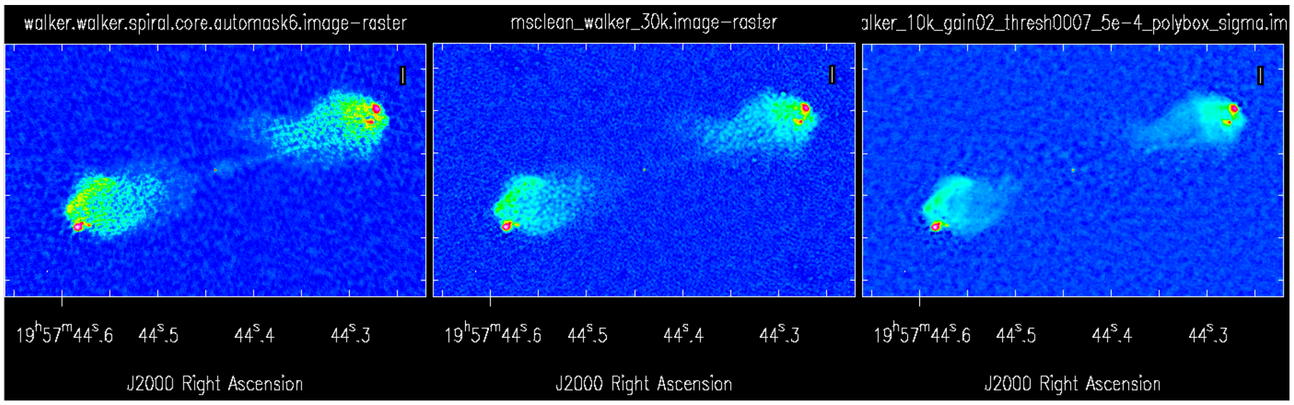

walker.walker.spiral.core.automask6.image-raster
msclean_walker_30k.image-raster
alker_10k_gain02_thresh0007_5e-4_polybox_sigma.im
19h57m44s.6
44s.5
44s.4
44s.3
J2000 Right Ascension

Figure 3.15: Restored image of the CygA complex model, with a 5min snapshot observations using the ngVLA Walker configuration: Högbom CLEAN with automasking vs. MS-Clean vs. Asp-Clean

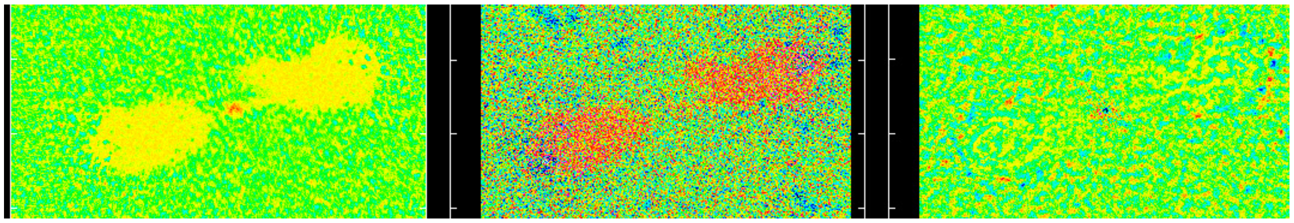


Figure 3.16: Residual image of the CygA complex model, with a 5min snapshot observations using the ngVLA Walker configuration: Högbom CLEAN with automasking vs. MS-Clean vs. Asp-Clean

3.6 ngVLA Imaging

The 8 GHz model for Cygnus A, processed as described in Carilli 2019 ngVLA memo 64, was used to simulate dataset as seen by ngVLA with the "Walker" configuration. This "complex" model has a size of $5" \times 2"$, or roughly 5×10^4 synthesized beam over the full source area. Figure 3.15 shows the restored images for the 5min snapshot observation of the complex model using the Högbom CLEAN with automasking, MS-Clean without automasking and Asp-Clean with polynomial masks. MS-Clean was run with three scales of sizes 0, 7 and 25 pixels and a very low loopgain (0.02). Both of the Högbom CLEAN and MS-Clean were ran for 30000 iterations. The Asp-Clean was ran for 20000 iterations. Snapshot imaging of a complex model is hard and and is typically seen as a ill-defined problem. The Asp-Clean imaging result has demonstrated this is possible with scale sensitive algorithms and the Asp-Clean algorithm is signal-to-noise ratio optimal. Asp-Clean residual is also more noise-like comparing with the Högbom CLEAN and MS-Clean (Figure 3.16).

Since the first verison of the memo, this test is updated to confirm the new optimization library can achieve better computational performance without degrading the imaging performance. Figure 3.17 shows that Asp-Clean with the original GSL (left) and the new ALGLIB (right) optimization libraries (Section 2.1.5) have similar imaging performance while the latter provides a 4x speedup.

3.7 EVLA Simulation

We use a simulation of a jet and lobe like structure to showcase a realistic situation, namely thin jets as well as a mix of compact and extended structure. The dataset has 5 channels, going from 1 GHz to 2 GHz for the VLA D-config. The truth values of the source spectral index can be derived from the smoothed model image cubes via a pixel by pixel spectral index fit. For the jet, the bottom spot had a flat spectrum and alpha goes from -0.2 to -0.8 from the jet to the lobe, with alpha of -1.0 for the spot at the top (Figure 3.18, left).

Figure 3.18 shows that the current MS-Clean solution (middle figure) cannot get the spectral index correct for the long edges of the jet part, and the Asp-Clean (right figure) has the spectral index much closer to the truth. Figure 3.19 shows

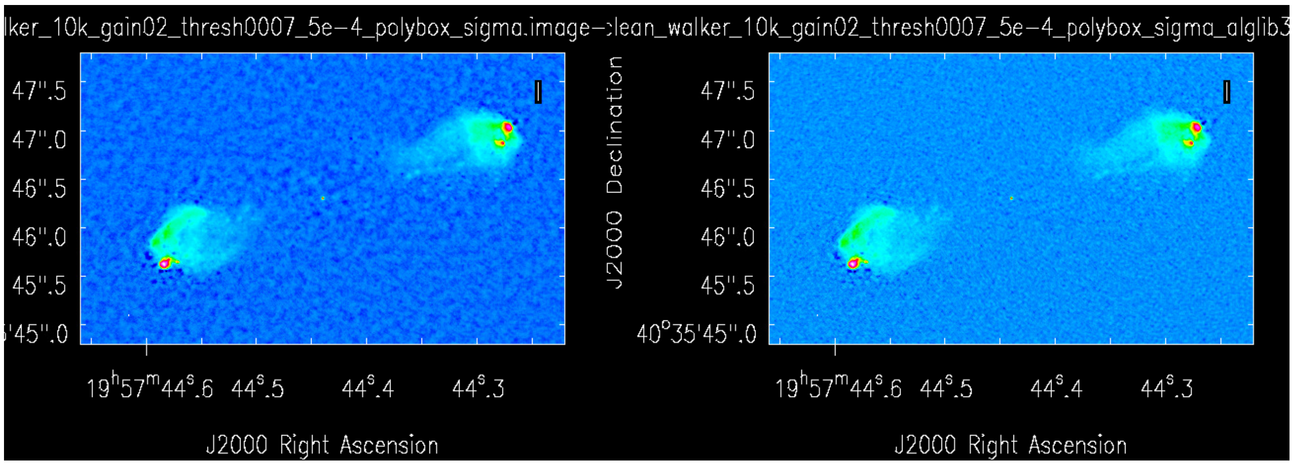

lker_10k_gain02_thresh0007_5e-4_polybox_sigma.image-clean_walker_10k_gain02_thresh0007_5e-4_polybox_sigma_alglib3
47".5
47".0
46".5
46".0
45".5
'45".0
40°35'45".0
J2000 Declination
19h57m44s.6 44s.5 44s.4 44s.3
J2000 Right Ascension

Figure 3.17: Restored image of the CygA complex model, with a 5min snapshot observations using the ngVLA Walker configuration: Asp-Clean with GSL vs. Asp-Clean with ALGLIB

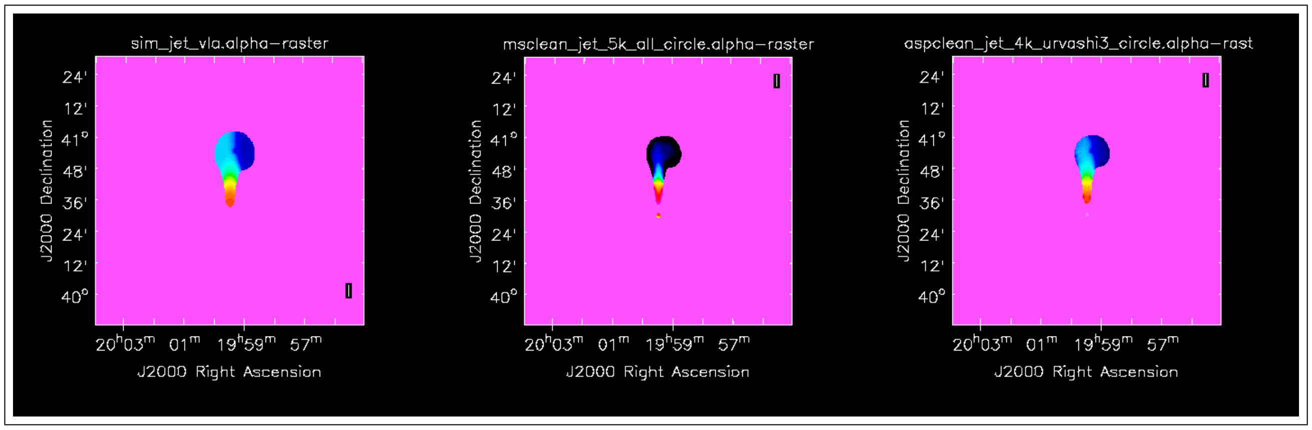


Figure 3.18: Spectral index comparison between the truth (left), MS-Clean (middle), and Asp-Clean (right) on the jet dataset.

the residual images for the MS-Clean (left) and Asp-Clean (right). The Asp-Clean residual images are more noise-like than MS-Clean. The new optimization library, ALGLIB (Section 2.1.5), was used for these Asp-Clean results.

3.8 Performance Comparison with Other CLEAN algorithms

We compared the memory consumption and and runtime between Asp-Clean and MS-Clean. In summary, Asp-Clean has better imaging performance and 25-40% less memory consumption but its runtime is about 6x longer comparing to MS-clean using multi-threading and about 5x longer comparing to MS-Clean without multi-threading. However, the runtime comparison is not based on comparable runs where in the case of CygA, MS-Clean was ran with 5 scale but Asp-Clean picked 285 scales. Moreover, Asp-Clean runs faster than the Högbom with automasking which is what the Production ALMA Pipeline uses. Note that the performance analysis results in this section are from the version of Asp-Clean when this memo was first written in early 2021 (i.e. with the GSL library and without the use of root mean square of residual to ”switch” to hogbom, see Chapter 4). Performance analysis of the new Asp-Clean updated in late 2021 is described in Section 3.8.

Figure 3.20 shows the histogram of the 285 scale sizes that Asp-Clean picked for the CygA dataset (Section 3.5). The interval of the bin size of the histogram is 5 pixels. We can see there is an outlier around 140 pixels. It was picked in an early iteration and was corrected in the later iterations. Besides, the four bars that are definitely higher than the others (i.e. scale sizes, 10, 20, 40, and 80 pixels) are very close to what we would manually set for the MS-Clean in addition to the

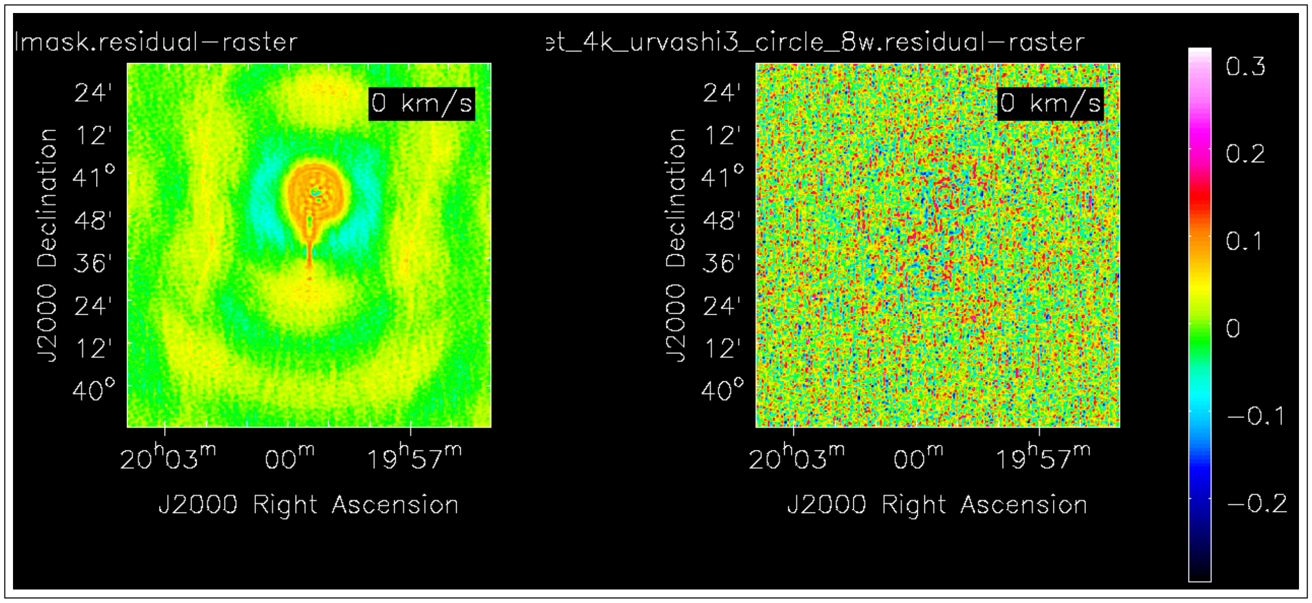


Figure 3.19: Residual image comparison between the MS-Clean (left) and Asp-Clean (right) on the jet dataset.

0 scale. Moreover, If we chose the scale sizes that have more than 4 occurrence to be the "dominant scales", based on this histogram the dominant scales = [10 15 20 25 30 35 40 50 60 70 75], total 11 scales. These are the "best case hand tuned" scale sizes for MS-Clean. The residual image of MS-Clean with the 11 scales is slightly better than the 4-scale one, but still not as good as Asp-Clean.

Another interesting observation is that when we changed the bin size of the histogram to $2*PSFWidth*beamwidthpixels$ (i.e. 1, 2, 4, 6, 8, 10, 12, 20), the histogram shows the logarithmic behavior that we assume when setting MS-Clean (see Figure 3.21).

Table 3.1 summarizes the runtime and convergence speed of Högbom with automasking, MS-clean without automasking, without multi-threading and Asp-Clean without automasking for the CygA dataset. We use the metric, number of iterations * loopgain, to measure how many iterations the algorithms take to get to the similar noise level, eliminating the effect of loopgain. In general, Högbom CLEAN needs a loopgain of 0.1, MS-Clean can have loopgain up to 0.5 safely, and Asp-Clean can easily have loopgain up to 0.8 (for ALMA M100_12m_7m and CygA) or 0.9 (for M31 and G055.7+3.4). In terms of computational efficiency, the runtime of Asp-Clean is only about 3x longer than MS-Clean with 11 scales (the best case hand tuned for MS-Clean) and Asp-Clean has better imaging performance. Therefore, we are not overly worried about the Asp-Clean runtime. Especially, we believe it will soon become significantly faster once we move it to GPU described in Chapter 4. Both MS-Clean and Asp-Clean run faster than Högbom with automasking. It is worth noting that these performance metrics are based on one dataset. Automasking time varies with its settings and more tests are required to fully characterize it.

Table 3.1: Performance Comparison of the CLEAN algorithms

	Högbom with automasking	MS-Clean (4 scales)	MS-Clean (11 scales)	Asp-Clean
runtime	7.5	1	1.64	4.64
convergence speed	2	1	1	1.4

Updated Performance Analysis of Asp-Clean in late 2021

The updated performance analysis on Asp-Clean was done by Intel Vtune. A brief summary is shown below.

- 70% of Asp-Clean runtime is on casacore::FFTServer::fft0 and flip. 30% is on matrix operations that are not localized in one function.

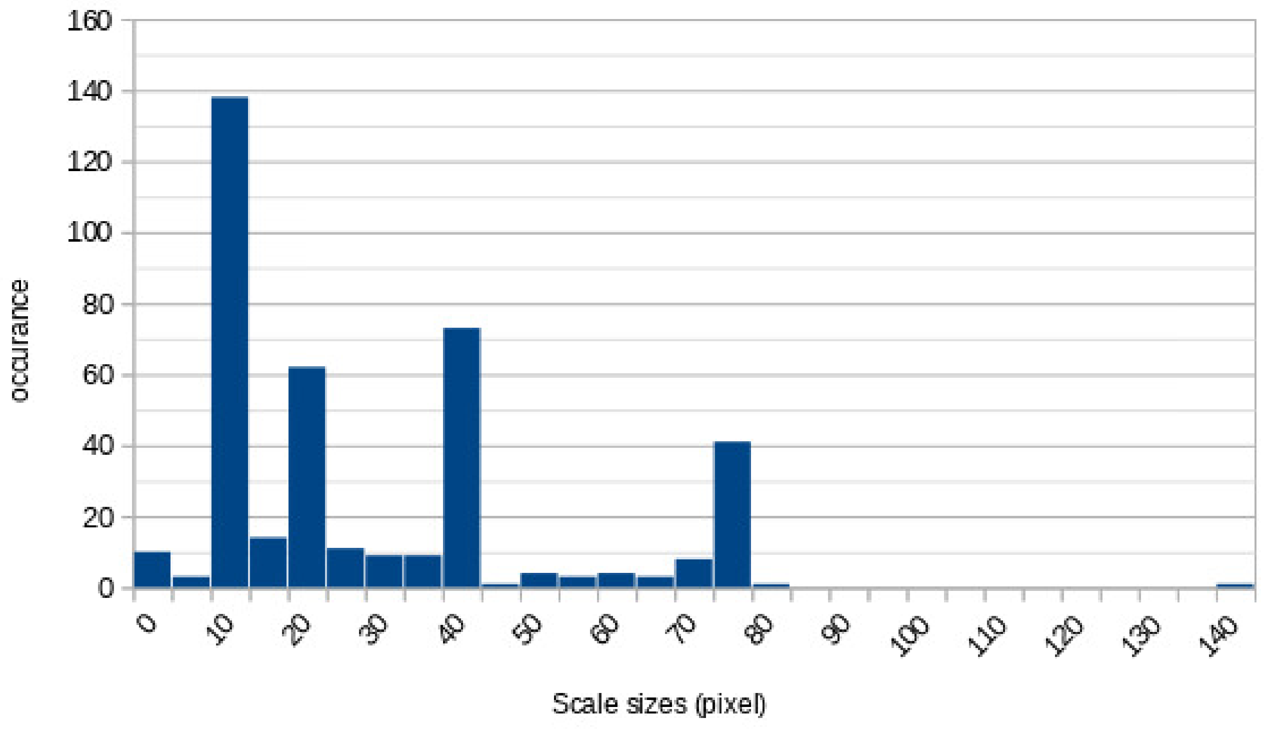


Figure 3.20: Histogram of scale sizes Asp-Clean picked for CygA

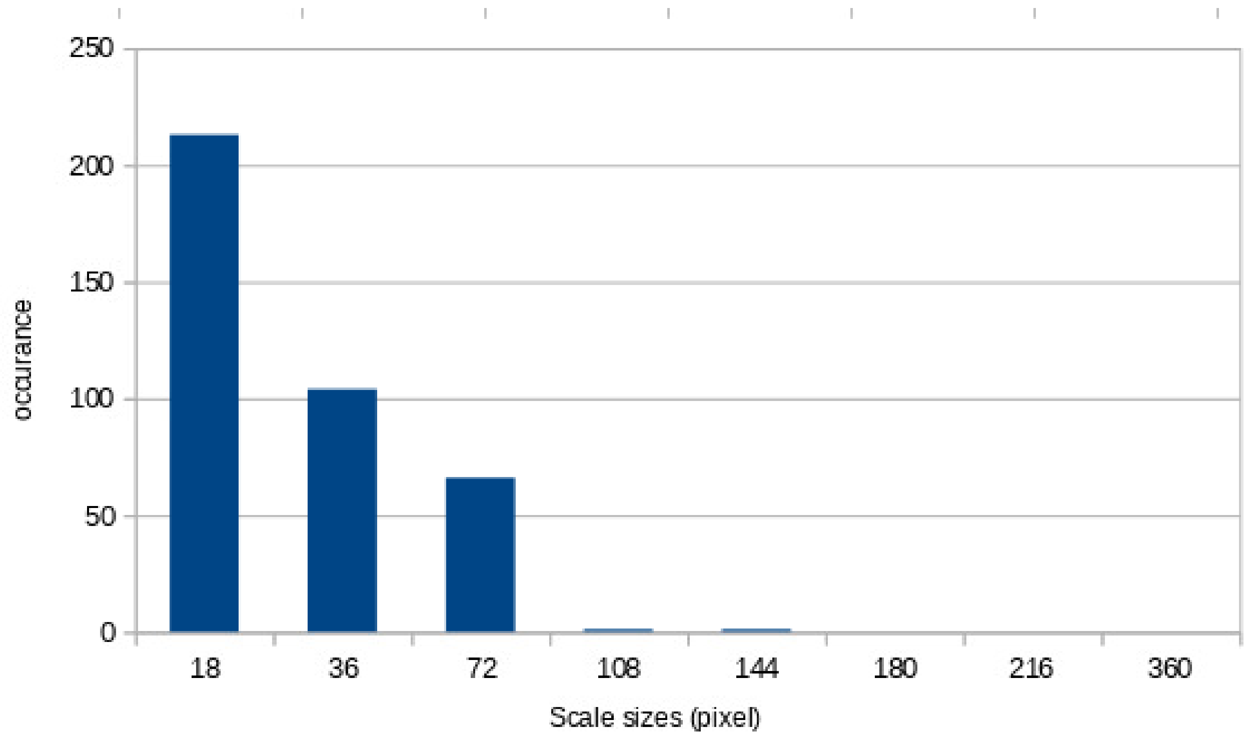


Figure 3.21: Histogram of scale sizes Asp-Clean picked for CygA shows the logarithmic behavior

- About 65% of Asp-Clean runtime is on the BFGS optimization. For the BFGS optimization, matrix calculation (17%) is considered cheap compared to fft0 (83%).
- Therefore, the performance bottleneck is at fft0/flip, which is not seen in MS-Clean. A GPU-based fft can improve the performance.

Chapter 4

Ongoing and Future Work

Our ongoing and planned work for Asp-Clean is summarized below.

1. Improvement on when to switch to Högbom CLEAN. Currently, we rely on what is described in Section 2.1.3 to trigger the switch, which mostly depends on manually determining a good threshold, `fusedthreshold`, to achieve a good balance of imaging performance and computational efficiency. This can be furthered improved by an automatic approach like the following.
   - Switch to Högbom CLEAN when large emission is modeled and subtracted. One way is to compute the auto-correlation of the residuals. It will be a centrally peaked function. If the width of the peak is comparable to the main lobe of the PSF, it will indicate that the larger scale emission is removed.
   - Pattern recognition method. Imaging processing technique is pretty matured to detect objects in an image, and in our case to detect if there is no more large emission in the residual image.

   Since the first version of the memo, an automated approach has been added to improve how Asp-Clean is switched to Högbom CLEAN. That is, when the peak residual has rarely changed (change is less than 1e-4 which is an experimental number) for two iterations, Asp-Clean is switched to Högbom CLEAN. Besides, if the root mean square of the residual is rarely changed, Högbom CLEAN is run for longer iterations before it is ”switched” back to Asp-Clean.
   We also experimented with using image analysis techniques to estimate the width of the peak component in the autocorrelation matrix of the residual image (AC width) and used that as the largest initial scale size. This was to experiment if we can use a better way to set the initial scales and to determine when the residual image is noise-like. We implemented this in a branch, and applied it on the ”jet” dataset. Our observations are the following.
   - For cube imaging, the AC width approach can be better, worse, or about the same as the current approach. It seems to work better for channels with higher frequencies. Asp-Clean with either approach is still much better than MS-Clean.
   - The runtime of the AC width and the current approaches are about the same.
   - From these observations, we do not see the advantages of using the AC width approach. This is probably due to imaging processing techniques can hardly determine the accurate AC width when the edges of the peak component is blurred.
2. GPU utilization to further improve computational efficiency. This will require R&D to conclusively determine the computing hot-spots in Asp-Clean implementation and exploring ways to deploy those on a GPU for speed up. We hope the 6x extra time mentioned in Section 3.8 will go down so Asp-Clean can become the most competitive algorithm for all kinds of applications.
3. R&D Wide-band Asp-Clean. Research will be conducted to develop algorithm that can use Asp-style modeling of the wide-band emission which will be built on the basic narrow-band Asp-Clean framework. The implementation will be tested with simulated and real data and comparing with existing wide-band algorithms.
   In late 2021, the implementation of the wide-band Asp-Clean (WAsp) has been completed and tested with various datasets. A paper is in preparation.
4. Research better Aspen models. Aspens are modeled by Gaussians right now (Eq. 2.1). Since they have wings that extend forever, they are are a good match to model sharp edges. They produce emission in the model beyond the

edge, and then the algorithm tries to correct it by putting negatives around the edges. This can be clearly seen in the Asp-Clean restored image in Figure 3.15. One idea is to use Higher Order Gaussians (HOG) whose shape can be controlled by its Gaussian expression.

Bibliography

Bhatnagar, S. & Cornwell, T. J. 2004, A&A, 426, 747

Bochkanov, S. 1999, ALGLIB, http://www.alglib.net

Cornwell, T. J. 2008, IEEE Journal of Selected Topics in Signal Processing, 2, 793–801

Högbom, J. A. 1974, A&AS, 15, 417

M. Galassi, M., Davies, J. T., Gough, B., et al. 2009, GNU Scientific Library Reference Manual - Third Edition (Network Theory Limited)

Qiu, Y. 2015, LBFGS++

Wieschollek, P. 2016, CppOptimizationLibrary, https://github.com/PatWie/CppNumericalSolvers

Zhang, L. 2018, A&A, 618, A117