User's Guide to Obit Software for the GBT/Mustang 3 mm Bolometer Array

W. D. Cotton, May 26, 2009

***Abstract*—This memo describes usage of the Obit software package to analyze data from the Mustang 3 mm bolometer array on the Green Bank Telescope. The various steps are given in some detail as well as a general discussion of the techniques. The processing uses a combination of python scripts and compiled programs. Several useful scripts are given in the appendix. The output of this process are images in FITS or AIPS format and calibrated OTF FITS data.**

***Index Terms*—single disk, bolometer array, imaging**

CONTENTS

National Radio Astronomy Observatory, 520 Edgemont Rd., Charlottesville, VA, 22903 USA email: bcotton@nrao.edu

Manuscript received ; revised

I. Introduction

MUSTANG, the 3 mm bolometer array on the Green Bank Telescope (GBT) observes in the "On-the-fly" (OTF) mode by scanning across the field to be imaged in a pattern intended to fully cover the region of interest. A general discussion of the techniques used are given in [1] [1]. Much of the following discussion is adopted from this reference.

This memo documents the usage of the Obit package [2] [2] to analyze Mustang data. An earlier version of this documentation is given in [3] [3].

This memo begins with an overview of the various aspects of analyzing Mustang data in Obit. Following this is a more detailed discussion of the various operations involved. Next is a short description of using Obit and ObitTalk. Finally the appendix gives the full text of a number of scripts as well as the detailed documentation for OTFSCal, the main Mustang iterative imaging and calibration program.

II. Overview

Obit consists of a c language library with python bindings and a set of compiled programs for specific operations. This allows both scripting for flexibility and tasks with well defined parameters. The current approach is to use a combination of these. The end products of this process are calibrated images in FITS or AIPS format and calibrated OTF FITS data.

The basic operations required are:

1) **Convert GBT archive to Obit OTF format**
   Obit has its own FITS based format for OTF data and the first step in processing is the conversion from the GBT archive format into OTF FITS format. As part of this translation the data is averaged, the 1.4 Hz from the pulse tube cooler removed and [optionally - NYI] short term pointing corrections can be applied based on the GBT Quadrant detector's measurement of the feed arm motion. This operation is performed with a python script.
2) **Examination and editing of data**
   On occasion, there are problems with the instrument, telescope or weather and it may be necessary to examine the time-stream data in detail. In the case of data with serious problems, it can be "flagged" - excluded from further consideration using the OTFFlag table in the OTF data-set. These operations are performed with a python script.
3) **Pointing calibration**
   The current observing technique defers determining and applying pointing correction to the off–line calibration. Pointing offsets need to be determined from calibrator sources and to be applied to the the data. This operation is performed with a python script.
4) **General calibration**
   There are a number of general calibrations that can be applied to an OTF data-set as a whole. These include, amplitude calibration to Jy, opacity corrections, applying pointing offsets, determining data validity weights and a removal of the gross detector background level. This operation is performed with a python script.
5) **Data adaptive background estimation**
   The detailed separation of the celestial signal from the background signals from other sources uses an iterative scheme in which the backgrounds and celestial signals are refined. This operation is carried out in Obit task OTFSCal.
6) **Flux density calibration**
   Variations in the efficacy of the OOF surface correction and uncertainties in the atmospheric opacity mean that each observing session needed to include observations of a source of known flux density. The observations need be imaged and used to calibrate target source images.

These are discussed in detail in section III

[1] ftp://ftp.cv.nrao.edu/NRAO-staff/bcotton/Obit/SDImage.pdf
[2] http://www.cv.nrao.edu/∼bcotton/Obit.html
[3] ftp://ftp.cv.nrao.edu/NRAO-staff/bcotton/Obit/MustangObit.pdf

III. Processing Steps

The following discuss the various processing steps in more detail; example python scripts are given in the appendix. These scripts can be executed using ObitTalk as described in Section VII.

A. Convert GBT archive to Obit OTF format

The GBT data acquisition system leaves data for a given experiment in a directory structure under /home/gbtdata on the GBT network. The subdirectory name is derived from the project code and session number, e.g. AGBT08A_056_12 is project AGBT08A_056 session 12. Each telescope subsystem logs its data in a separate subsystem, one file per scan.

These files need to be parsed and converted into a coherent data-set for further processing. This is done using a ObitTalk (python) script that looks into the archive and decided which scans were observed and uses Obit task PAROTF2 (for 2009–? data) to append each scan to the output OTF data file. In this process, the data are time averaged, the 1.4 Hz pulse tube signal (see section V-F3) is fitted and removed, and short period pointing errors due to the feed arm can be corrected using the GBT Quadrant detector (this feature is not yet functional). The GBT archive and OTF formats are described in more detail in Section VI. A sample script for reading archive format data and writing an OTF file is given in the appendix in Section A. This script needs a configuration file to describe the detector offsets, the file for the 2009 season is in $OBITSD/share/data/PAROTF09.cfg. More details can be obtained in ObitTalk by :

```
>>> import GBTUtil
>>> help(GBTUtil.UpdateOTF).
```

B. Examination and editing of data

Time-stream data can be viewed in great detail using the fv utility (Section VI-A) but this is generally not a tractable approach to examine large quantities of data. ObitTalk has facilities for generating plots that allow systematic exploration of a data set and allow applying selected calibration, editing and data selection criteria. An example script is shown in the

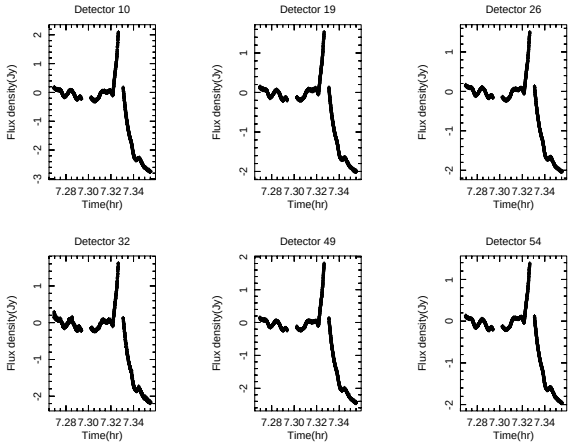

Fig. 1. Example OTF data plot for several detectors.

appendix in Section B, the output of which is shown in Figure 1.

If there are times with poorly behaved data, such as the large jump seen in all detectors in Figure 1, it may be desirable to flag sections of data to exclude them from analysis. This is done by making entries in the OTFFlag table in the OTF data-set describing the data to be flagged. An example of an ObitTalk script to do this is in the appendix in section scriptFlagData

C. Pointing calibration

The current observing practice is not to determine or apply telescope pointing corrections during the observations. Therefore, these corrections must be determined and applied in the off–line calibration. These pointing offsets can be determined from the in focus scans on the bright calibrator sources used for focus and surface corrections. The script in the appendix in Section D has internal documentation of its use. This script will write output data and images in directory FITSdata which should be created before running the script. This script can be run on the calibrators from an observing session producing a set of images and at the end of the script producing the python code fragment that can be inserted in the the general calibration script:

```
PointTab=[
    [ 0.29765,   -1.35,   -6.83], # ...
    [ 0.30910,   -1.18,   -7.40], # ...
    [ 0.31132,   -2.41,   -6.79], # ...
    [ 0.33364,   -2.07,   -7.97], # ...
    [ 0.35168,   -2.78,   -3.98], # ...
    ]
```

D. General calibration

Once an OTF data-set has been generated, any questionable data flagged, and pointing corrections determined, the data file can be calibrated. An example of an ObitTalk script to do this is in the appendix in section E; a template version of this script is in $OBITSD/share/scripts/scriptMustangCali.py The value of python array PointTab in this script needs to be replaced by the values derived by the pointing calibration described in Section III-C. The calibration operation will generate a series of "Solution" (OTFSoln) tables and "Calibration" tables (see Section VI) which describe how to calibrate the data.

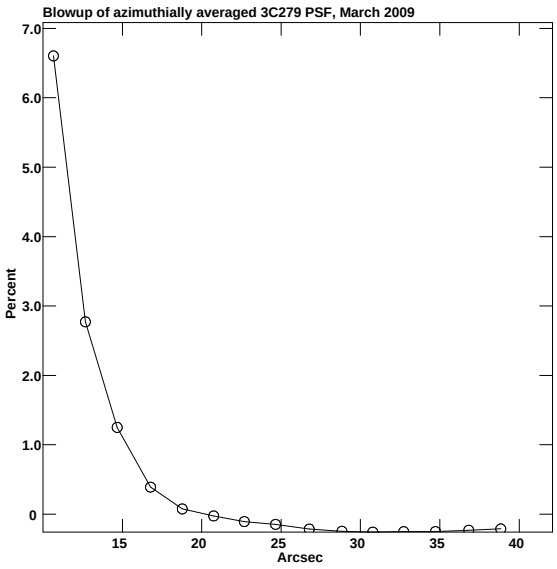

Fig. 2. Azimuthally averaged GBT PSF..

E. Data adaptive background estimation

Once data editing and general calibration is completed, the data adaptive calibration is done using the OTFSCal program run from ObitTalk. This program iteratively estimates the sky model and the backgrounds as described in Section V-G.

1) Telescope PSF: The OTF calibration and imaging program OTFSCal takes an image which is the instrumental PSF. Since OTFSCal does not make parallactic angle dependent corrections, the instrumental PSF should be azimuthally symmetric. The telescope PSF is not to be confused with the image produced of an unresolved source as this image has been convolved with the griding function.

The suggested technique to obtain the effective PSF is to use the calibrators used for the target. This can be done by making a calibrated image using 4x over sampling (cell spacing of 0.5") using a "pillbox" griding function (ConvType=0 in OTFSCal) and then discard 3 of each rows and columns, keeping the pixel centered on the peak. Then, normalize the image to a unity peak and azimuthally average by averaging in rings centered on the peak. This can be done with AIPS task IRING. Then, create a PSF image using a Gaussian of 8" FWHM to 2 cells from the center after which interpolate the azimuthally averaged calibrator image. This can be performed using a script similar to the one in the appendix in Section F. This script was used to create a PSF from 3C279 observations in March 2009 which is available in $OBITSD/share/data/PARPSFv2.fits.gz. A blowup of the azimuthally averaged image is shown in Figure 2.

2) Running OTFSCal: Program OTFSCal reads an input OTF data files and creates an image and a calibrated output OTF file for the list of targets given. This program separates

the time domain signals into “sky” and “background” by a iterative process in which the shortest timescale of the background signal is reduced as the model of the celestial sky improves.

OTFSCal is a compiled program which is most easily run from the ObitTalk interface. In order to use the interactive features of OTFSCal, ObitView should be started prior to starting ObitTalk., see Sections VII-B and V-D. From ObitTalk, a task interface object can be created:

```
>>> otfscal = ObitTask("OTFSCal")
```

Note, the name of the interface object, here “otfscal” is arbitrary. Once this this object is created, it can be used to examine the parameters to be passed to the program,

```
>>> otfscal.i
```

view the on-line documentation,

```
>>> otfscal.h
```

or execute the program

```
>>> otfscal.g
```

Parameters needed to run OTFSCal can be specified as follows:

```
# Input OTF data file name
otfscal.inOTF = "AGBT09A_052_06OTF.fits"
# Input GBT PSF file name
otfscal.PSFFile = "PARPSFv2.fits"
# Specify Output files
otfscal.outOTF = "TargetCalOTF.fits"
otfscal.outFile = "Target"
# List of targets
otfscal.targets = ["target"]
# Prior Calibration/editing parameters
otfscal.doCalib = 1
otfscal.gainUse = 0
otfscal.flagVer = 1
# Imaging parameters
otfscal.xCells = 2.0
otfscal.yCells = 2.0
otfscal.nx = 200
otfscal.ny = 200
# Clean parameters
otfscal.CLEANBox = [-1,20,100,100]
otfscal.Niter = 5000
otfscal.BeamSize = 8.0
# Iterative calibration parameters
otfscal.commonInt = 4.0
otfscal.solnInt = 0.5
otfscal.solnMult = [4.0, 2.0]
```

Program usage and parameters are described in the appendix in Section G; most parameters have sensible defaults which need not be explicitly set. Note, FITS “disk” 0 indicates that the files are expected and will be written in the current directory. Examination of intermediate results and modifying the CLEAN box is possible using ObitView. Processing parameters are written in the “History” tables on the output FITS or AIPS data-sets.

The more critical and useful options are discussed in the following, parameters are given in `this font`.

- **Initial calibration with a prior image**
  An initial sky model can be given and the initial “background” signals estimated from the residuals to this model. This mode is enabled by specifying the FITS file in parameters `priorFile` and `priorDisk`. If `priorMod` is True, then the CLEAN components table (“AIPS CC”) on `priorFile` is convolved with `PSFFile` and used as the sky model model; otherwise the pixels in the image are used as the sky model. The residuals from this model subtracted from the time-stream data are “common” mode filtered with shortest timescale `priorInt`. This calibration is applied in all subsequent iterations.
- **Initial calibration with common mode filter**
  An initial flat (zero) sky model can be used as the initial calibration. This mode is invoked if `commonInt` is $>$ 0.0. `commonInt` is the shortest timescale in the common mode background estimation. This calibration is applied in all subsequent iterations.
- **Ignoring wild data points**
  Data points outside of the range $\pm$ `clipData` are ignored in forming the images.
- **Cleaning**
  CLEANing is the stage of decomposing the raw image into a set of delta functions. The maximum number of delta function is `Niter`, the CLEAN loop gain is `Gain`, and the minimum residual in the CLEAN is `minFlux`. CLEAN is restricted to pixels inside the CLEAN window which can be specified in `CLEANBox` or interactively using ObitView. The derived sky model will be zero outside of the CLEAN box and the delta functions convolved with the PSF image.
- **Iterative background estimation**
  Multiple loops of background estimation can be specified using `solnInt` and `solnMult`. `solnInt` is the time increment in the calibration tables and `solnMult` specifies a sequence of calibration stages in which the residuals from the previous sky model (CLEAN, prior, common mode or no model) are common mode filtered to shortest timescale `solnMult[i]` times `solnInt`. The sequence `solnMult[*]` should monotonically decrease; a minimum of 2.0 is fairly conservative and should never be less than 1. In each calibration, detector offsets will be estimated if `doOffset` = True and the timescale will be `offsetFact` times that of the common model timescale.
- **Limiting range of sky model values**
  In deriving the value of the sky model used for background calibration, the range of pixel values can be limited to `minResFlx` to `maxResFlx`.
- **Data editing on comparison with sky model**
  if `doEdit` is true, then after the final cycle of background estimation, the residual time-stream data is examined for large and variable residuals. In each `flagInt`, each

detector time-stream is compared with the model time-stream and data with an RMS in excess of max(`maxRMS`, `maxRatio`*model_RMS) is flagged, bad time segments are listed in OTFFlag table `flagver` and applied in the final imaging

F. Flux density calibration

Flux density calibration depends on variable conditions such as opacity and how well the OOF corrects the surface shape. Calibrate by imaging point sources of known flux density using the same imaging parameters as were used for the target.

IV. Use Cases

The following sections give some suggestions for analyzing Mustang data in a number of different cases. Limitations and cautions are also given.

A. Strong, small, isolated sources

This is the simplest case for Mustang observations; common mode filtering, even on a fairly short timescale should work well. In OTFSCal, use `commonInt` of a couple seconds as the initial calibration. The CLEANbox should be specified relatively closely to the source but large enough to include the shoulder on the PSF. The zero level is effectively set by the region in the image outside of the CLEAN window. Calibration timescales as short as 0.5 second appear practical.

B. Weak, small sources

If there is no extended emission in the field, process as for a strong source but use `minResFlx` and `maxResFlx` to restrict the range of allowed values in the sky model from of order -sigma to somewhat above the expected peak.

C. Strong extended, sources

In this case, the most difficult aspect of the imaging is correctly setting the zero level; setting the CLEAN window is fairly critical in this. Note, however, that the calibration will attempt to remove any emission outside of the CLEAN box. Setting `minResFlx` to a slightly negative value helps remove the negative bowls that common mode filtering tends to put around extended sources. For sources which are larger than the footprint of the array on the sky, it may be useful do do a single additional iteration in which the CLEAN model from one run of OTFSCal is used as the prior image in the subsequent imaging.

D. Weak extended source

Imaging weak extended sources is similar to that for strong sources except that setting the range of allowed sky model values is more critical (`minResFlx` and `maxResFlx`). A tighter CLEAN window also helps at the potential expense of removing true emission.

The griding function can be used to trade sensitivity against resolution. Using the default Gaussian function (`ConvType`=3, `ConvParm`=[0.,0.,0.]) gives a resolution with Mustang of about 10” but with better sensitivity.

An initial common mode with or without a prior model with a `commonInt` or `priorInt` of a few seconds can really help with the initial model. There is a trade-off between the timescale of this calibration and the size of the CLEAN box and the ability to recover very extended structure. Structure very much larger than the array footprint on the sky can easily be absorbed into the background calibration and removed. On the other hand, too large a value of `commonInt` or `priorInt` and or the CLEAN box will allow atmospheric or instrumental fluctuations to be mapped into large scale structure which may persist in the processing. `minResFlx` can be used to prevent spurious negative emission but this still allows spurious extended positive features.

E. SZ in clusters

This is like weak extended sources in general except that the `maxResFlx` may be slightly positive; or little more than any emission in the field. Using the Gaussian griding function is highly recommended. Calibration timescales shorter than about 1 second may not be useful.

V. Description of Technique

The following sections describe the general calibration and imaging technique in some detail.

A. Calibration Approach

The principle difficulty with imaging short wavelength radio continuum single dish data is the large and variable background signal. This background comes from the atmosphere, the telescope and the instrument itself and can vary on a wide variety of timescales. The offset of the measured signals from zero is particularly difficult to determine and generally must be determined from some knowledge of the (astronomically interesting) sky. The basic approach in Obit Single Dish software (ObitSD) is to iteratively model the background signals, image the sky, subtract a model of the astronomical sky from the data and re-estimate the backgrounds. The estimate of the background is determined from the residual data (sky model subtracted) by a low pass filtering. As the quality of the model improves, higher temporal frequency components of the residuals are included. This technique uses the redundancy in the data to separate the constant astronomical sky from the variable background signals. The zero level is set essentially by the constraint that the sky brightness in some regions imaged are zero. This technique works best if the telescope beams can be swept over the field of view sufficiently fast that the modulation of the signal due to the beam sweeping over the astronomical sky is faster than the time scale of the variations of the background signals.

The redundancy in the data are of two forms. The first is that each resolution element in the image is covered multiple times by the instrument, hopefully along different trajectories on the sky as in the “basket–weaving” technique. These multiple observations of each piece of the sky help separate the constant portion of the signal from the time variable component.

For instruments with multiple pixels, such as the Mustang bolometer array, there is additional information. The contributions of the atmosphere will be essentially the same in all detectors, i.e. a common mode signal, as the various beams strongly overlap through the regions of the atmosphere in which the bulk of the brightness variations occur. In the case of Mustang, much of the background variation from the instrument itself (e.g. the 1.4 Hz variations due to the refrigerator pump) are also common mode. For sources smaller than the detector array, these common mode backgrounds are easily distinguished from the source signals which appear in only a few pixels.

The implementation of the ObitSD calibration is an iterative one. The OTF data is kept in a single FITS file and the calibration is manipulated with tables of a pair of types; a "solution" (OTFSoln) table and a total "calibration" (OTFCal) table. The structures of these tables are identical and contain a number of multiplicative and additive (both per detector and common) and other corrections. The usage of the two types is different. A "solution" table is an incremental set of calibrations, usually derived from data after the application of a total calibration table. A new total calibration table is then derived by the application of the solution table to the previous calibration table. This new total calibration can then be applied to the data. In addition to the calibration tables, there is a "flagging" table (OTFFlag) which can be used to describe data to be ignored (actually given zero weight). More details of these tables are described in section VI.

B. On the Fly (OTF) Imaging

In the "On-the-Fly" imaging technique, the telescope is swept across the field of view in a pattern that will cover all of the field to be imaged while sampling data at a constant rate. The data sampling should be fast compared the the time it takes the beam to move its own width or the sky will be smeared out. In this mode, the sky is sampled at uniform times but at positions not constrained to the pixels on a well defined grid. As this is the case, the actual position on the sky of each detector must be accurately known at all times.

The conversion of the "randomly" sampled data to a regular grid proceeds in a number of steps collectively called "griding".

1) "Convolution" and re-sampling on a grid
   Each sample is considered as a delta function and is multiplied by a continuous "griding" (AKA "convolution") function. The griding function is centered on the datum and is sampled on a grid centered on the pixel nearest to the datum and which is of finite support. This results in a grid of data*griding_function as well as a grid of griding_function values.
2) Accumulation onto a grid
   The data times the griding function and the griding function samples are accumulated onto a pair of grids covering the region of the sky to be imaged.
3) Normalization
   When all of the data have been multiplied by the griding function, re sampled and accumulated onto the grids, the image is normalized by dividing the sum of the data*griding_function grid by the sum of the griding_function grid on a pixel-by-pixel basis.
4) deMode [optional]
   If the bulk of the pixels in an image are expected to have no emission, this can be enforced by using the deMode option. The mode of the pixel distribution is determined, this is presumed to be the actual background level which should be zero, and the value of the mode is subtracted from all pixels in the image.

In general, not all data are of the same quality, e.g. varying sensitivity among detectors, and each datum may be assigned a statistical weight. These weights are included in the process described above by replacing the griding function with the product of the griding function times the weight of the datum being gridded.

In the griding procedure, the griding function plays a critical role and the function chosen is generally a compromise. One of the compromises is sensitivity against resolution. Using a "fatter" griding function will include more data in each pixel and thus give lower noise at the cost of reduced resolution.

The resolution of the derived image affects the units which, by convention, are expressed in Jansky per beam area. Any operation that modifies the resolution (beam area) must be reflected in the units of the derived image. In general, this requires a scaling by the ratio of the beam areas.

Obit currently provides the following griding functions as separable functions on a rectangular grid, generally the default parameter values are adequate.

- **Pillbox**
  The pillbox function is 1 inside the area of the pixel and 0 outside. The following parameters are allowed:
  - Parm[0] = half-width in cells of support [def 0.5]
  - Parm[1] = Expansion factor
- **Gaussian**
  This is a Gaussian centered on the central image grid cell. The following parameters are allowed:
  - Parm[0] = half-width in cells of support [def 3.0]
  - Parm[1] = Gaussian with as fraction of raw beam [def 1.0]
- **Exponential*Sinc**
  The following parameters are allowed: This is as exponential times a Sinc (sin x/x) centered on the central image grid cell. The following parameters are allowed:
  - Parm[0] = half-width in cells of support [def 2.0]
  - Parm[1] = 1/sinc factor (cells) [def 1.55]
  - Parm[2] = 1/exp factor (cells) [def 2.52]
  - Parm[3] = exp power [def 2.0]
- **Spheroidal wave function**
  This is a prolate spheroidal wave function centered on the central image grid cell. The following parameters are allowed:
  - Parm[0] = half-width in cells of support [def 3.0]
  - Parm[1] = Alpha [def 5.0]

C. Telescope beam-shape and deconvolution

Even with the traditional and OOF holographic corrections the GBT beam-shape at 90 GHz still has a pedestal containing a non trivial fraction of the total power. The details are likely time variable but the pedestal persists as a general feature of the telescope beam. If the actual telescope point spread function (PSF) is known, it can be taken out in the CLEAN deconvolution. In practice, the PSF is known only approximately. A further complication is that even a stable PSF will rotate on the sky with parallactic angle. The simple CLEAN currently being used assumes a circularly symmetric telescope PSF, requiring another level of approximation. The presence of a pedestal around main lobe affects extended emission and the region around a bright point. A description for making a usable GBT PSF image is given in Section III-E1 and one is shown in Figure 2.

D. Using CLEAN to Derive Sky Model

In order to use the method outlined in Section V-A, a sky model must be derived from the image obtained from the data. The use of a griding function in the image formation in general has a point spread function (PSF) larger than the intrinsic resolution of the telescope so is not directly useful. In addition, there are artifacts in the image resulting from imperfect background estimation that should not be included in the sky model.

The sky model used in ObitSD is that obtained from a CLEAN deconvolution of the image derived from the data. This has the advantage that the CLEAN components can be restored with an approximation of the actual telescope beam, meaning that the telescope response can then be determined from a simple interpolation of the CLEAN restored image. In addition, CLEAN windowing can be used to constrain the region in which emission is allowed. In the CLEAN sky model, the residuals are not included which gives the region outside of the CLEAN window zero flux density. CLEAN windows can be either hard-coded into the processing script, set interactively or use Obit's automatic windowing algorithm. A screen-shot of an interactive window setting session using ObitView is given in Figure 3

If the CLEAN sky model is derived from an image with resolution different from the telescope, the units of the sky model image must be scaled to those appropriate for the telescope data. Such resolution changes are inherent in the OTF imaging process.

E. Background Signals

Telescope radiometers, and especially bolometers, measure the power that hits their detectors. This power comes from a number of sources in addition to that from the celestial object being observed. In addition to the background sources of power, variations in the instrument itself will cause variations in the output time stream data which mimic variations in received power. The following describes several of the major backgrounds and other sources of spurious signals and general strategies for reducing their effects.

One of the major uncontrollable sources of background is the variable emission from water vapor in the troposphere. On the largest scales, the emission varies with the air-mass through which the sky is being observed; for sufficiently large fields being imaged, the variation across the field of view can be significant at any one time. For a constant, known zenith opacity, correction as a function of observing air-mass is straightforward.

On smaller scales, water vapor cells have scale sizes of order a hundred meters and can introduce significant brightness variations on time scales of seconds to minutes. Since the emission is largely very close to the antenna, the lines of sight from the various detectors to the sky mostly pass through the same volume of air so this background is essentially common mode among the various detectors. For sources much smaller than the footprint of the array on the sky, subtracting a running median of the detector values from each data stream largely removes the atmospheric contribution. For sources which are not small compared to the footprint, it is desirable to sweep the telescope beams over the source at a sufficiently high speed that the modulation of the celestial signals is faster than the variation of the atmospheric signals.

Most wide-band detectors suffer from "1/f" noise at some level; this is caused by low level gain fluctuations whose power spectrum is "red" meaning larger fluctuations on longer timescales. Removal of this background is also helped by a rapid motion of the telescope beams over the sky to help separate the instrumental from the celestial signals in the time domain. Repeated, frequent re-observation of the same position of the sky also helps constrain this background.

Other instrumental variations can also modulate the bolometer detector outputs in ways which may-or-may-not equally effect all detectors. One such common mode signal is the 1.4 Hz variations from the pulse tube refrigerator on the MUSTANG array.

F. MUSTANG Array

MUSTANG (Multiplexed SQUID TES array at Ninety GHz) [4] is a 64 element bolometer array operating at 90 GHz on the Green Bank Telescope (GBT) giving about 8" resolution. MUSTANG was commissioned during the winters of 2006–2007 and 2007–2008 and has begun preliminary science operations.

1) Observing Patterns: The instrument observes by scanning rapidly over the field to be imaged with the bolometer data streams sampled at a constant, 1 kHz rate. The current observation modes are "daisy" and "boxscan" patterns; both are continuous scanning patterns. The former makes repeated passes through the center of a circular scan region and the latter uniformly covers a rectangular regions with a pattern of nearly orthogonal passes.

2) Calibration: Amplitude and gain calibration is performed using an internal calibration source that is cycled between the "on" and "off" states. The data are calibrated by comparing the response to this calibration source to that of a planet of known brightness. Partial daisy patterns on bright quasars are used for focus and pointing measurements. Focus

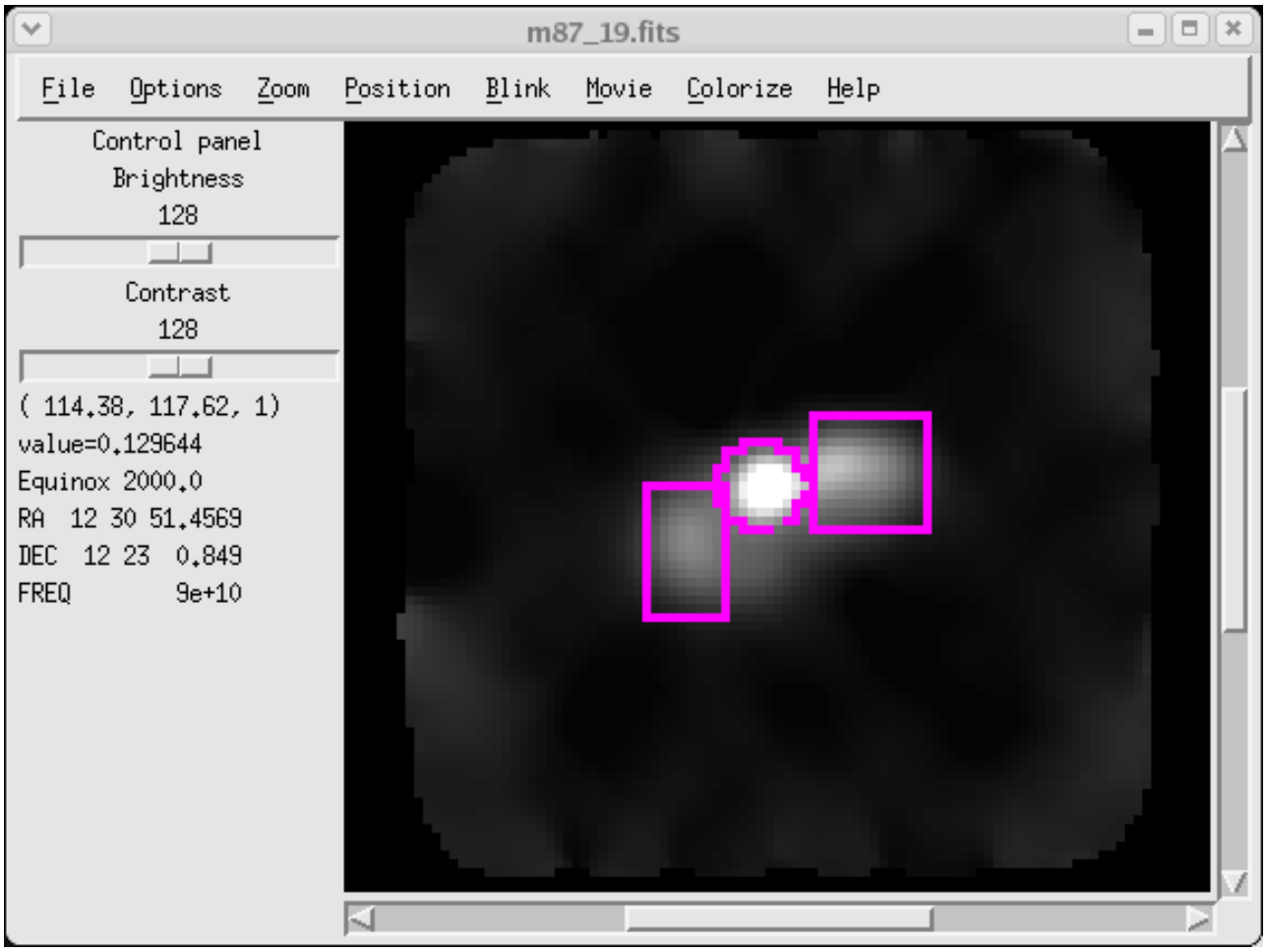

Fig. 3. Screen-shot of interactive CLEAN window editing using ObitView.

corrections are applied during the observations but pointing corrections are applied off–line. Currently a constant zenith opacity of 10% is used. The weather in Green Bank is seldom better than this; if it is much worse than this, the data is of questionable use.

3) Correction of Internal Oscillations: The MUSTANG bolometer detectors are very sensitive detectors of power in any form and are sensitive to mechanical power from the pulse tube cooling mechanism. This introduces a very narrow band modulation of the detector outputs from all detectors; an example power spectrum is shown in Figure 4. Since this is a very strong signal, it is easily estimated and removed from the time–stream data. A simple notch filter approach is inappropriate as this will also remove power from the sky signal at this frequency introducing artifacts. This is especially the case when there are strong gradients in the sky brightness such as around bright point sources. It was empirically determined that this signal has a very narrow intrinsic width and is centered near 1.41170 Hz. A sine wave at this frequency with common phase and detector dependent amplitude was fitted to, and removed from, the detector time-stream data for each scan. A sample of data before and after this filtering is shown in Figure 5.

G. Iterative Background Estimation Concepts

In the following discussion the contributions to the detector outputs are considered to be of two types; 1) response to the celestial sky which we desire to estimate and 2) “backgrounds” which are everything else. The principle difficulty with imaging short wavelength radio continuum single dish data is the large and variable background signal. The basic approach described here is to iteratively model the background signals, image the sky, subtract a model of the astronomical sky from the data and re-estimate the backgrounds. The estimate of the background is determined from the residual data (sky model subtracted) by a low pass filtering. As the quality of the model improves, higher temporal frequency components of the residuals are included. Any prior knowledge of the sky (e.g areas with no emission) can be introduced as constraints on the sky model.

1) CLEAN: A CLEAN deconvolution is used to reduce the image of the sky to a set of delta functions. Ths set of delta functions if the sky model.

2) Subtraction of Sky Model: The response of the telescope to the CLEAN model of the sky is given by the convolution of this set of delta functions with the telescope PSF. Note: in general this is different from the PSF of the “dirty” image subjected to the CLEAN. The CLEAN delta functions after

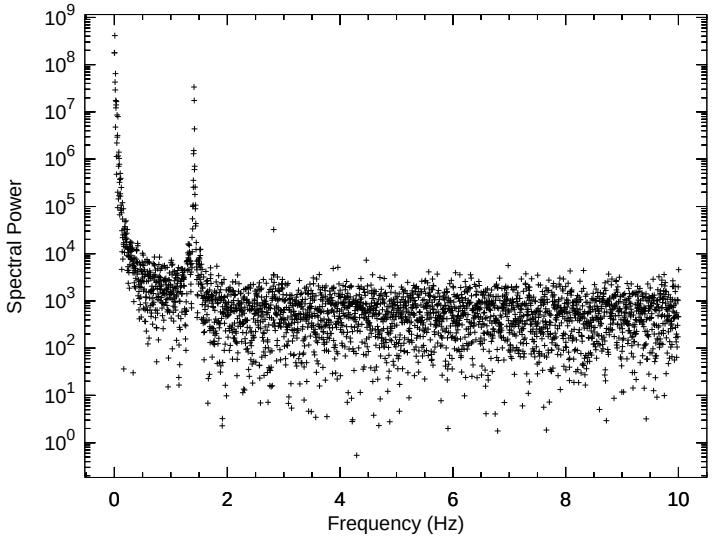

Fig. 4. The power spectrum of a segment of output data from one element of the MUSTANG array on the GBT. The sharp increase towards the lowest frequencies is characteristic of both the atmospheric and "1/f" noise contributions. The 1.4 Hz modulation due to the pulse tube cooler is also prominent.

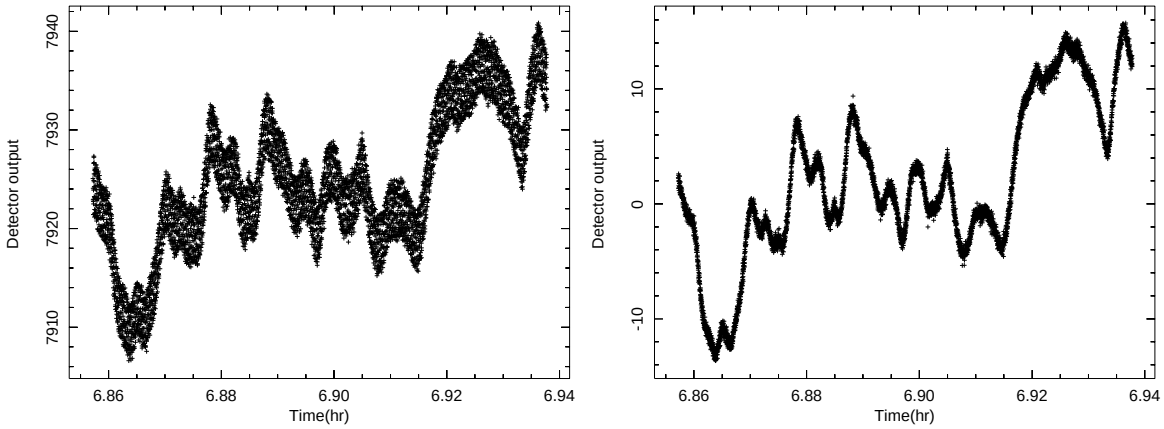

Fig. 5. The left plot shows the time sequence of 5 minutes of 20 Hz averaged data from a single MUSTANG detector including the 1.4 Hz signal as a function of time. The right plot shows the same data after fitting and subtracting the 1.4 Hz signal.

convolution with the instrumental PSF and scaling into the appropriate units, can have any additional physical constraints imposed (e.g. positivity) and used to derive the instrumental response to the sky model by simple interpolation. To produce a residual data set, the instrumental response to the model is evaluated at the position in the sky model of all data samples and subtracted from those samples.

3) Backgrounds from filtered residuals: To the degree that the sky model is a perfect representation of both the true sky and the telescope PSF, the residual data should contain only the background signals and the time series representing the backgrounds can be derived from a filtering of the residual data. Since the data sampling rate is significantly faster than variations in either the response to the sky and variations in the background, this filtering should be a low pass filter. Early in the iteration, the sky model will be imperfect and the filtering should pass only temporal frequencies in the residual data long compared to the time it takes the telescope beam to pass over structures in the sky. Then, as the sky model improves, higher temporal frequencies can be included in the estimated background.

4) Common mode vs. detector backgrounds: Background signals come from a number of different causes, those associated with individual detectors (e.g. gain variations) will be largely independent from detector to detector. Those backgrounds which are common to all detectors (e.g. tropospheric emission) will have approximately equal effects on all detectors and can be considered common mode effects. The detector specific and common mode effects may occur on different timescales and can be determined separately.

H. Imaging and Background Estimation

The outline of the background estimation algorithm is shown in Figure 6. After an initial set of calibrations based on external measurements, an iterative refinement of the sky model and the backgrounds is performed on the data themselves. As the accuracy of the sky model improves from cycle to cycle, the filter timescale (τ) is reduced until it approaches that of the actual background; for the MUSTANG array this is usually of the order of one second as can be seen in Figure 4. Since the detectors themselves exhibit variations, the background estimated includes both that of individual

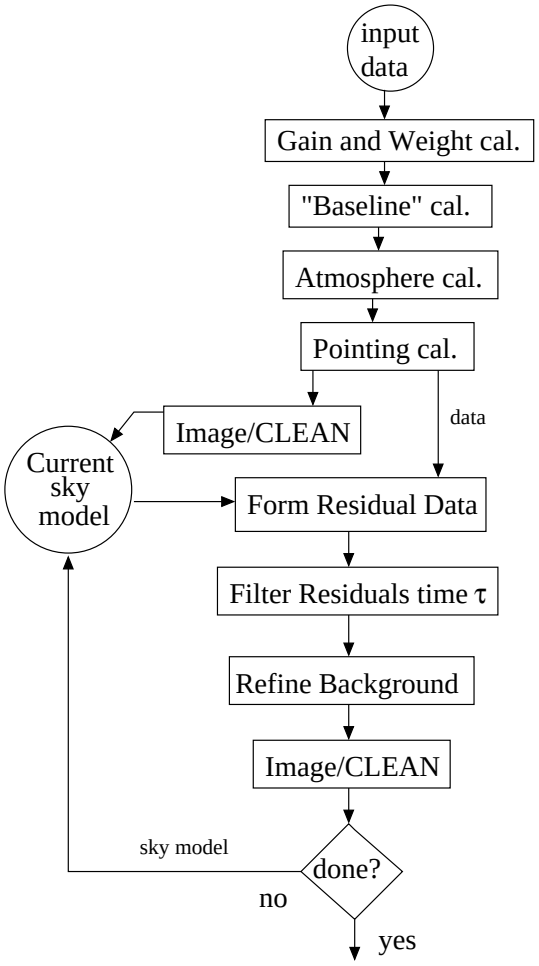

Fig. 6. Flow diagram of the iterative background estimation process. Each iteration the timescale, τ, of the filtering is reduced. Background type can alternate between common mode and detector. See the text for an explanation of each box.

detectors and the common mode background. Timescales for estimation of detector offsets are longer than for common mode backgrounds. Following are explanations of the boxes in Figure 6.

- **Gain and Weight Calibration**
  The gain of each detector is determined from scans with a cyclic firing of the MUSTANG internal calibration source. The equivalent source strength in Jy is determined by comparing the strength of the response to the internal calibration source to that of a strong source (planet) of known brightness. The strength of the calibration source is determined by subtracting the average "Cal-off" signal prior to and following the "Cal-on" signal. This signal is used to both stabilize the detector gains and convert the signal strengths into Janskys. Not all detectors are equally sensitive and the relative weights are determined from the RMS fluctuations in short scans of blank sky measurements.
- **"Baseline" calibration**
  The "baseline" for each detector in each scan is estimated using a low pass filter on the data. The time scale from this filtering needs to be long enough not to remove significant power from any astronomical signals.
- **Atmospheric calibration**
  For multi-beam systems such as MUSTANG, the atmospheric contributions to the background are expected to be largely common among all the detectors. After the application of the previous calibration, a single offset for each detector and a piecewise linear common mode atmospheric offset series in time is robustly estimated. The timescale of the linear common mode segments is sufficiently long not to significantly impact the astronomical portion of the signal. Gain corrections are also made for the estimated zenith opacity and the air-mass of each data sample.
- **Pointing calibration**
  Corrections to the telescope pointing are made by linear interpolation of the pointing errors in azimuth and elevation as determined from the focus/pointing scans on nearby calibrator sources.
- **Image/CLEAN**
  After application of the best available calibration, the data are imaged as described in section V-B and a sky model formed using CLEAN as described in section V-D. This sky model image has the PSF of the telescope.
- **Form Residual Data**
  The sky model is interpolated to the position of each data sample and subtracted from the calibrated data-set to produce a residual data-set.
- **Filter Residuals**
  The residual data set is subjected to a low–pass time filter to remove timescales shorter than a time constant τ.
- **Refine Background**
  The filtered residual data are re-sampled to produce a refined estimate of the time variable background. This background estimate can be either per detector or a common mode estimated from the median detector value. The timescale τ used for detector offsets is longer than that used for common mode background.

I. Extended Emission

There is a fundamental degeneracy between celestial structures larger than the footprint of the array on the sky and the background signals. This degeneracy can be partially alleviated by very rapid motions of the telescope to modulate the sky signal faster than the backgrounds are varying and separate the effects in the time domain. This process can be greatly assisted if even imperfect knowledge of the sky brightness model can be used. Using the iterative background estimation technique, sky models derived removing only slow background variations are used to prevent large scale structure from being absorbed into the backgrounds estimated on shorter timescales.

In principle, this technique can be applied to a larger scale of iteration and repeating the entire sequence of background estimation by starting the process with the final sky model of a previous processing. Excessive use of this technique has its difficulties as well as it is possible to invent spurious large scale structure as well as remove it. Experience suggests a single reprocessing starting with the previous model help

recover most of the extended emission (largely eliminates the bowls around large sources) but the effects of further iteration have not been explored.

VI. OTF Data Format

The GBT archives data on a per scan basis with each logical component of the system (e.g antenna controller and receiver) logging its own data into separate FITS files. For use in the ObitSD OTF package, these files are read and converted into a single FITS file in a form similar to a relational database. This format is described in detail in ftp://ftp.cv.nrao.edu/NRAO-staff/bcotton/Obit/ObitSD.pdf. The data are kept in a single table with antenna pointing positions interpolated from the Antenna FITS files together with other information. Separate tables contain auxiliary information such as target information, offsets of each detector from the antenna pointing and a scan index. In addition, calibration and editing can be specified as tables which can be applied as the data are read. The standard tables are summarized in the following:

- OTFScanData
  Raw sky brightness data, pointing and other time variable information. A row in this table corresponds to all data at a given time.
- OTFArrayGeom
  Table giving the geometric offsets of a feed/detector array from the pointing axis of the telescope.
- OTFTarget
  Table of sources or targets. These are referred to in the OTFScanData table as an index into this table.
- OTFIndex
  Scan table [optional] giving start and stop times and row numbers in the OTFScanData table as well as targets etc. This index is used to improve data access times.
- OTFFlag
  Table describing “flagged” data - data to be ignored.
- OTFCal
  Cumulative calibration table. This table gives multiplicative and additive corrections to the raw sky brightness measurements in the OTFScanData table as well as corrections to the nominal telescope pointing direction.
- OTFSoln
  Differential calibration (“Solution”) table.

A screen-shot of fv displaying an OTF FITS file with data and calibration tables is shown in Figure 7.

A. Examining OTF Data Using fv

The plotting facilities in Obit are currently limited (see section III-B or in ObitTalk, import PARCal; help(PARCal.PlotData)) and the most general way to view Mustang data is using the fv utility from GSFC. A display of a segment of data from a number of detectors is shown in Figure 8.

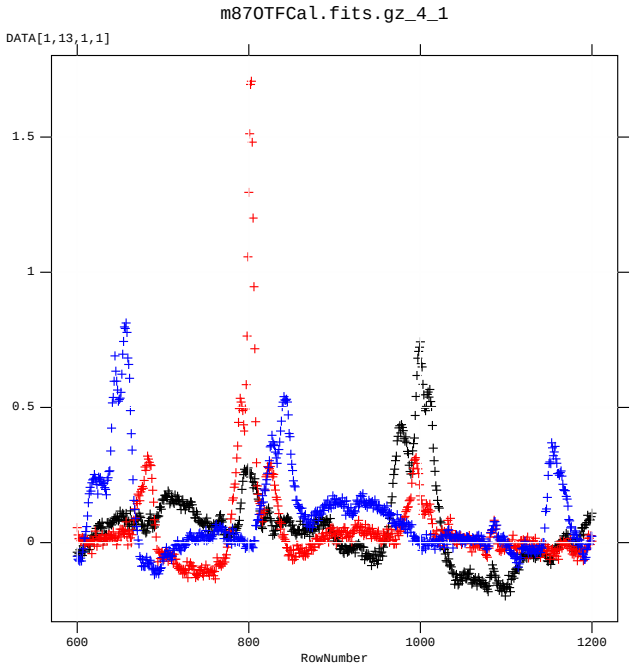

Fig. 8. fv display of a section of data from several detectors.

VII. Obit

A. Obtaining Obit Software

Obit and related software is available from http://www.cv.nrao.edu/∼bcotton/Obit.html. At present there is a system of stable releases but the Mustang software changes sufficiently rapidly that a stable release rapidly becomes out of date. Therefore, using the anonymous Subversion (SVN) interface for the current development is recommended. Obit depends heavily on third party software which is described on the Obit web page. Support of the Obit package is limited. The components of the Obit/ObitTalk package are:

- Obit
  Basic Obit package and the support for images and radio interferometry.
- ObitSD
  Obit “On The Fly” (OTF) single dish imaging package.
- ObitView
  Image display used by Obit.
- ObitTalk
  Scripting and interactive interface to Obit software.

These software packages come with installation instructions and config scripts to build them.

B. Starting ObitTalk

The Obit interactive and scripting python interface is ObitTalk which is a python interpreter with basic Obit packages preloaded. For an installation using the Obit installation, use the setup.sh or setup.csh scripts generated by the installation script in the root directory of the Obit installation to set these variables. An extensive guide to using ObitTalk is [5] [4].

If you wish to use the ObitView image display, you can start it before ObitTalk. If ObitView is in your path:

```
% ObitView &
```

[4] ftp://ftp.cv.nrao.edu/NRAO-staff/bcotton/Obit/ObitTalk.pdf

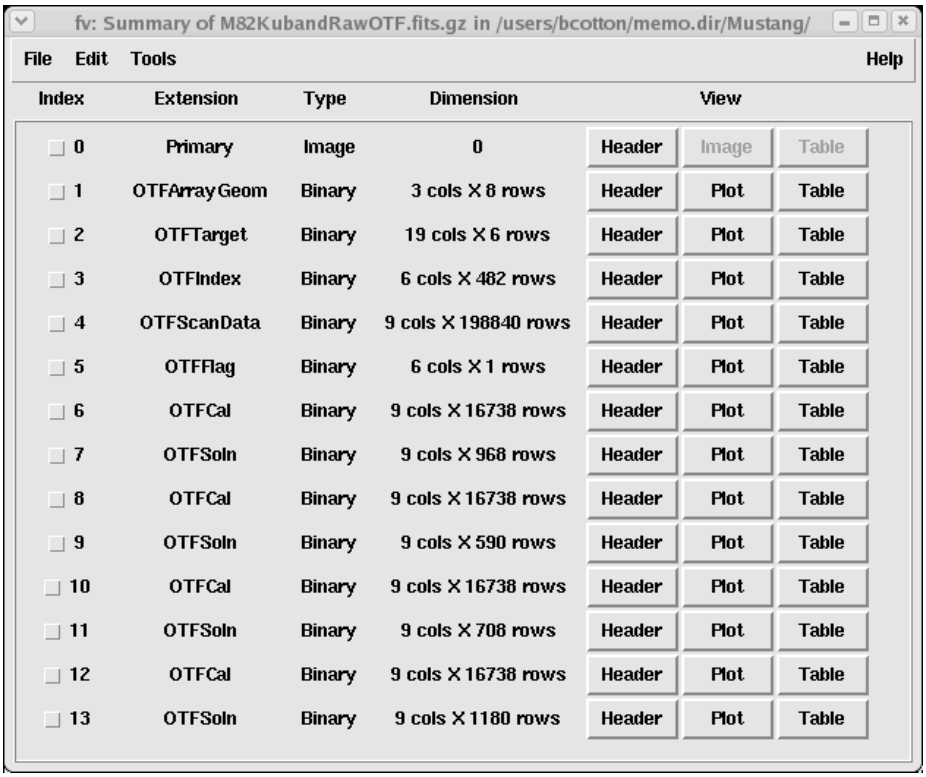

Fig. 7. Screen-shot of the fv display of an OTF FITS file containing data, calibration and editing tables.

will start the display server. If this fails to start the display, see the discussion of ObitView in the ObitTalk User Manual.

Then, if the script ObitTalk is in your path:

```
% ObitTalk [scriptname]
```

should start ObitTalk. If the environment variable AIPS_ROOT is defined, ObitTalk will make AIPS tasks and data available. If the optional script name is given, then the python interpreter will do some simple AIPS initialization and execute the python script “scriptname”. If no script is specified then ObitTalk will ask for your AIPS number and do its AIPS initialization (if AIPS is available) and go into an interactive python session. Alternately, AIPS information can be placed in an ObitTalk configuration script. If you are not planning on using AIPS, give any number in response to the request for an AIPS user number. The python prompts are:

```
>>>
```

.

C. Quick Guide to ObitTalk

Python is a basically “Object–oriented” language and much of the ObitTalk software follows this methodology. “Object–oriented” in this context means little more than variables are more substantial than that floats and strings and simple arrays (although these also exist). A python (hence ObitTalk) variable is a relatively arbitrary thing and can be a scalar number, string, an array or list of variables or the interface to a data-set such as an image or OTF data. In addition, functions which operate on the object can be attached to the object.

In ObitTalk, the interface to a data set is assigned to a variable and this variable is used to specify operations on that data-set. Similarly, the interface to a task (compiled program executed outside of the python interpreter) is an object with parameters and functions as members.

The usual object–oriented syntax is that “class methods” (functions which can operate on an object) are invoked like this:

```
>>> object.function(arguments)
```

where “object” is the python object, “function” is the function name and “arguments” are the additional arguments, the object is implicitly an argument, by convention called “self” in python. In python documentation of function interfaces, “self” appears as the first argument of the function although it is invoked as shown above. Note: many Obit functions have objects as explicit arguments, these use the form:

```
>>> module.function(object, other_arguments)
```

where module is the name of the module defining function.

Before a module can be used, it must first be imported into python. This is done using the python “import” command:

```
>>> import OTF
```

to import the basic OTF package. The documentation on that package can then be obtained:

```
>>> help(OTF)
```

for the entire package or

```
>>> help(OTF.AtmCal)
```

for a non class function or for a class function:

```
otf=OTF.OTF("otf")  # An object of type OTF
>>> help(otf.Open)
```

The following modules are of interest to single dish imaging:

- **OTF -** OTF (”On the Fly”) data
- **OTFDesc -** OTF data descriptor (header)
- **OTFUtil -** OTF Utilities
- **OTFRec -** OTF data record class
- **OTFGetAtmCor -** OTF Atmospheric correction utilities
- **OTFGetSoln -** OTF calibration solution utilities
- **OTFSoln2Cal -** Utilities to convert OTF solution to calibration tables
- **GBTUtil -** GBT OTF Utilities
- **CCBUtil -** GBT CCB utility package
- **CleanOTF -** Single dish (Hogbom) CLEAN
- **GBTDCROTF -** Convert GBD DCR data to OTF format
- **PARCal -** Mustang calibration/editing utilities
- **History -** History class
- **Image -** Image class
- **ImageDesc -** Image Descriptor (header)
- **ImageMosaic -** Image Mosaic class
- **ImageUtil -** Image utilities
- **InfoList -** Obit associative array for control info
- **ODisplay -** Interface to ObitView display
- **OErr -** Obit message/error class
- **OPlot -** plotting interface
- **TableDesc -** Table descriptor (header) class
- **TableList -** Table list for data object (Image, UVData, OTF)
- **Table -** Table class
- **TableUtil -** Table utilities
- **History -** History class

APPENDIX

A. Example script to read data

```
# Read Mustang data
import OSystem, OErr, GBTUtil

err     = OErr.OErr()     # Obit error/message object
FITS    = ["./"]          # OTF data in curernt directory
user    = 100             # AIPS user number (anything will do)
ObitSys = OSystem.OSystem  ("ReadMustang", 1, user, 1, ["None"], 1, FITS, \
                            True, False, err)
OErr.printErrMsg(err, "Error with Obit startup")

################################################################################
# Define parameters

# Root of data directory
DataRoot="/media/SimpleDrive/Mustang09/AGBT09A_052_06/AGBT09A_052_06/"

# Define data
# output OTF file
inFile   = "AGBT09A_052_06OTF.fits"  # Output OTF file name
inDisk   = 0                         # 0 => current working directory
config   = "../PAROTF09.cfg"         # Gives detector offsets

################################## Update data ################################
# Update OTF with any new scans in archive
inOTF = GBTUtil.UpdateOTF("PAROTF2","Rcvr_PAR",inFile, inDisk, DataRoot, err, \
                          config=config)
OErr.printErrMsg(err, "Error creating/updating input data object")

# Shutdown Obit
OSystem.Shutdown(ObitSys)
del ObitSys
```

B. Example script to plot data

```
# Plot selected OTF data
import OTF, PARCal, OErr

err = OErr.OErr()

# Get OTF data file data
otf = OTF.newPOTF("data", "AGBT08C_026_02OTF.fits", 0, True, err, nrec=1)

# Want calibration
otfList = otf.List
otfList.set("doCalib", 1)
otfList.set("gainUse", 2)              # Only gain calibration
otfList.set("flagVer", 1)              # Apply any prior flags
otfList.set("timeRange", [0.,10.])

# Specify data to plot
targets = []                                      # Any Target name
feeds   = [10,19,26,32,49,54]                     # 1-rel feed numbers
nx      = 3; ny=2;                                # make 3x2 plots per page
scans   = [32,34]                                 # scan range
```

```
# Output file name
output = "dataPlot"+str(scans[0])+".ps/ps"

# plot
PARCal.PlotData(otf, targets, scans, feeds, err, output=output, nx=nx, ny=ny)
print "plot to "+output
```

C. Example script to flag data

```
# script to flag bad data
import OTF, OSystem, OErr, OTFUtil

# Init Obit
err     = OErr.OErr()     # Obit error/message object
FITS    = ["./"]          # OTF data in curernt directory
user    = 100             # AIPS user number (anything will do)
ObitSys = OSystem.OSystem  ("flagPAR", 1, user, 1, ["None"], 1, FITS, \
                            True, False, err)
OErr.printErrMsg(err, "Error with Obit startup")

# Define data
disk    = 0                          # 0 => current directory
inFile   = "AGBT08A_056_12OTF.fits" # OTF data file

# Set data
inData = OTF.newPOTF("Input data", inFile, disk, True, err)
OErr.printErrMsg(err, "Error creating input data object")

# Flag bad data
chans      = [1,0]          # all channels
flagVer    = 1              # Flagging table 1
stokes     = "1111"         # Stokes flag (all)
reason     = "Bad"          # reason string

# Flag all data for detectors 36,37,45 all
flagtimer = [0.0,1000.0]
feeds = [36,37,45]
target = "Any"
for feed in feeds:
    OTFUtil.PFlag(inData,err,flagtimer,target,flagVer,chans,stokes,feed,reason)

# Flag detector 52 from time 8.710 to 8.713 (hours)
flagtimer = [8.710/24.,8.713/24.]
feed      = 52                         # Detector 52
OTFUtil.PFlag(inData,err,flagtimer,target,flagVer,chans,stokes,feed,reason)

# All data from 8.965 to 8.970
flagtimer = [8.965/24.,8.970/24.]
feed      = 0                          # All detectors
OTFUtil.PFlag(inData,err,flagtimer,target,flagVer,chans,stokes,feed,reason)

# Shutdown
print ’Done: flagged data in’,inFile
OSystem.Shutdown(ObitSys)
del ObitSys
```

D. Example script to determine pointing corrections

```
# Determine pointing correction from calibrator observations
import Obit, OErr, OSystem, OTF, Image, ODisplay
import PARCal

# Init Obit
err     = OErr.OErr()     # Obit error/message object
user    = 100             # AIPS user number (anything will do)
# directory where output images are to go
FITS    = ["/export/data_2/fits/Mustang09/AGBT09A_052_06/FITSdata"]
ObitSys = OSystem.OSystem  ("MustangPoint", 1, user, 1, ["None"], \
                            len(FITS), FITS, True, False, err)
OErr.printErrMsg(err, "Error with Obit startup")

PointTab = None  # No previous pointing corrections

# list of [source_name,list of scan numbers to include] per cal image
scanList = [
    ["1415+1320",5], ["1415+1320",28], ["1415+1320",33,34], ["1415+1320",45],  \
    ]

# extract target list
target = []
for scan in scanList:
    target.append(scan[0])

# Calibration parameters
CalJy = [38.5]                  # Cal values in Jy, one for all or per detector
BLInt  = 20.0                   # Baseline filter time in sec
AtmInt = 10.0                   # Atmospheric filter time in sec
solInt = 0.5                    # Min. solution time
tau0   = 0.1                    # Zenith opacity

flagver  = 1                         # Apply any flagging
gainuse  = 0                         # Apply highest numbered OTFCal table
inFile   = "AGBT09A_052_06OTF.fits"  # Full Data set
BeamFile = "../PARGaussBeam.fits"    # Dirty Gaussian beam (PSF)
inDisk   = 0                         # 0 means current working directory

# OTF file
inData = OTF.newPOTF("Input data", inFile, inDisk, True, err)
OErr.printErrMsg(err, "Error creating input data object")

# dirty beam
dirtyBeam  = Image.newPFImage("dirty beam", BeamFile,  inDisk,  True, err)
OErr.printErrMsg(err, "Error initializing dirty beam")

# Initial calibration
timerange=[0.0,10.0]              # time range in days, All times
inInfo = inData.List
inInfo.set("timeRange", timerange)
PARCal.InitCal(inData, target, err,\
               flagver=1, CalJy=CalJy, BLInt=BLInt, AtmInt=AtmInt,\
               tau0=tau0,PointTab=PointTab)
OErr.printErrMsg(err, "Error with initial calibration")
```

```
# Set pointing calibration parameters
timerange=[0.0,10.0]              # time range in days, All times
inInfo = inData.List
inInfo.set("timeRange", timerange)
input = PARCal.FitCalInput
input["InData"]   = inData
input["PSF"]      = dirtyBeam
input["solInt"]   = solInt    # minimum Solution interval
input["gainUse"]  = 0
input["Niter"]    = 200       # Number of CLEAN iterations
input["scrDisk"]  = 1
input["scanList"] = scanList  # List of scans
input["disp"]     = ODisplay.ODisplay("display","ObitView",err)
input["save"]     = True   # Keep images
input["ConvType"] = 4      # Exp*Sinc

# Image/fit
result = PARCal.FitCal(err, input)
OErr.printErrMsg(err, "Error with calibration")

# Print results
print "result",result
rr = []           # Table of corrections
for r in result:
    rr.append([r["Time"],-r["azOff"], -r["elOff"], r["Target"], \
               r["Gauss"], r["Peak"]])
    print r

# Print pointint table in python format
print "pointing\nPointTab=["
for r in rr:
    print "    [%8.5f, %7.2f, %7.2f], # %20s, Beam=(%5.2f, %5.2f, %6.1f), Peak=%5.2f \\ "\
          %(r[0],r[1],r[2], r[3],r[4][0],r[4][1],r[4][2],r[5])
print "    ]"


# shutdown Obit
OErr.printErr(err)
del ObitSys
```

E. Example script to calibrate OTF file

```
# Script for bulk calibration of Mustang data
# This script applies file global calibrations which do not depend on
# a source model.
# Calibration tables left on the OFT data file.

import OSystem, OErr, InfoList, History
import OTF, PARCal, Obit

# Init Obit
err     = OErr.OErr()     # Obit error/message object
user    = 100             # AIPS user number (anything will do)
FITS    = ["./FITSdata"]  # Where to put output/scratch files
ObitSys = OSystem.OSystem  ("MustangCali", 1, user, 1, ["None"], \
                            len(FITS), FITS, True, False, err)
OErr.printErrMsg(err, "Error with Obit startup")

################################################################################
# Define parameters
# Define data
# OTF file
inFile   = "AGBT09A_052_06OTF.fits" # OTF data file
#           ^^^^^^^^^^^^^^^^^^^^^^     SET THIS
inDisk   = 0                        # 0 means current working directory

# Default Calibration info
flagver    = 1                      # Flag table to apply
CalJy      = [38.5]                 # Cal values in Jy, one for all or per detector
BLInt      = 30.0                   # Baseline filter time in sec
AtmInt     = 20.0                   # Atmospheric filter time in sec

# Table of pointing offsets in time order [time(day) d Xel (asec), d el (asec)]
PointTab=[\
    [0.0, 0.0, 0.0], \
    [1.0, 0.0, 0.0]]
#    *********** SET THIS ********
# Pointing table generated by pointing script
PointTab=[
    [ 0.29765,   -1.35,   -6.83], # 1337-1257, Beam=( 8.43,  7.70,   31.8), Peak= 8.26 \
    [ 0.30910,   -1.18,   -7.40], # 1337-1257, Beam=( 9.27,  7.99,   43.0), Peak= 6.54 \
    [ 0.31132,   -2.41,   -6.79], # 1415+1320, Beam=( 9.91,  8.74, -137.1), Peak= 0.65 \
    [ 0.33364,   -2.07,   -7.97], # 1415+1320, Beam=(10.24,  7.97, -298.1), Peak= 0.68 \
    [ 0.35168,   -2.78,   -3.98], # 1415+1320, Beam=( 9.74,  8.47,   51.0), Peak= 0.61 \
    ]

# Table of opacities in time order [time(day), zenith opacity(nepers)]
tau0 = [[0.0000,    0.100],       \
        [1000.0000, 0.100]]
#    *********** SET THIS if needed ********

################################### Initial calibration ############################
PARCal.InitCal(inOTF, ["    "], err,\
               flagver=flagver, CalJy=CalJy, BLInt=BLInt, AtmInt=AtmInt,tau0=tau0, \
               PointTab=PointTab)
```

```
##################################### History ########################################
print "Add history"
# Loop over dirty, clean images, output data
outHistory = History.History("out history", inOTF.List, err)

# Add this programs history
outHistory.Open(History.READWRITE, err)
outHistory.TimeStamp(" Start Obit "+ObitSys.pgmName,err)

# Calibration
outHistory.WriteRec(-1,ObitSys.pgmName+" BLInt = "+str(BLInt),err)
outHistory.WriteRec(-1,ObitSys.pgmName+" AtmInt = "+str(AtmInt),err)
outHistory.Close(err)

# Copy history to header
inHistory  = History.History("in history",  inOTF.List, err)
outHistory = History.History("out history", inOTF.List, err)
History.PCopy2Header(inHistory, outHistory, err)
OErr.printErrMsg(err, "Error with history")

# Shutdown Obit
del ObitSys
```

F. Example script to generate a GBT PSF

```
# Make PSF using Gaussian inner core, to 4 asec and iring beyond
# Use fixed, average beam
# Beam will be azimuthally  symmetric

import Image, OErr, OSystem, FArray, ImageDesc, math
import FArray, FInterpolate

# Init Obit
err=OErr.OErr()
ObitSys=OSystem.OSystem  ("MakeGaussBeam", 1, 103, 1, ["None"], 1, \
                          ["/export/data_2/fits/Mustang09/AGBT09A_052_04/MakeBeam"], \
                          True, False, err)
OErr.printErrMsg(err, "Error with Obit startup")

# Parameters
beamFile = "PAR1DBeam2009.fits"  # Output FITS file
outDisk  = 0               # Output disk number
diameter = 90.0            # Illuminated diameter
centFreq = 90.0e9          # Center frequency in Hz
clight   = 2.997924562e8   # speed of light m/s
BW       = 16.0e9          # Bandwidth in Hz
cells    = 2.0             # Image cell spacing in asec
ncells   = 256             # Dimension of the beam as calculated
nsave    = 101             # Dimensionality of the beam saved (should be odd)


def BeamShape (offset, inFI, gau, flux, FWHM, err):
    """ Interpolate pixel value

    return the beam for a specified offset from the reference position
    Use a Gaussin inside gau and interpolate using inFI outside
    offset  = offset in pixels
    inFI    = Python Obit input FInterpolate
              First value should be @ gau
    gau     = Maximum extent of the Gaussian
    flux    = flux for Gaussian
    FWHM    = FWHM (pixels) for Gaussian
    err     = Python Obit Error/message stack
    """
    ################################################################
    if offset>gau:
        x = offset - gau + 1.0
        val = FInterpolate.P1D(inFI,x,err)
    else:
        sigma = FWHM / 2.3548
        val = flux * math.exp(-(offset*offset)/(2.0*sigma*sigma))
    return val
# end BeamShape

# Use Gaussian to 4 arcsec
# Fitted Flux = 10.6 FWHM = 11.64 = 5.82 pixels
```

```
# Shape of beam, every 2 asec from 4" = 2 pixels
# 2009 24 March 3C279
bshape=[0.5574, 0.3127, 0.1538, 0.6603E-01, 0.2771E-01, 0.1250E-01, 0.3896E-02, 0.7568E-03, \
        -0.2524E-03, -0.1064E-02, -0.1470E-02, -0.2149E-02, -0.2458E-02, \
        -0.2579E-02, -0.2514E-02, -0.2505E-02, -0.2311E-02, -0.2118E-02, -0.1830E-02, \
        -0.1510E-02, -0.1325E-02, -0.1154E-02, -0.1059E-02, -0.9107E-03, -0.8042E-03, \
        -0.7290E-03, -0.6927E-03, -0.6671E-03, -0.6546E-03, -0.6230E-03, -0.6183E-03, \
        -0.6426E-03, -0.6545E-03, -0.6750E-03, -0.6802E-03, -0.6989E-03, -0.7210E-03, \
        -0.7442E-03, -0.7360E-03, -0.7117E-03, -0.6915E-03, -0.6967E-03, -0.7076E-03, \
        -0.6986E-03, -0.7041E-03, -0.7501E-03, -0.7883E-03, -0.7933E-03, -0.7966E-03, \
        0.0]

# Make FArray
naxis= [len(bshape)]
fa=FArray.FArray("Beam1D",naxis)
i = 0
for b in bshape:
    b /= 0.8847
    fa.set(b,i)
    i += 1

# Dummy ImageDesc
id = ImageDesc.ImageDesc("dummy")

# Interpolator
fi = FInterpolate.FInterpolate("interpolator",fa,id,2)

# Gaussian parameters
flux = 1.0; FWHM = 4.0; gau=2.0  # Values in pixels

# Make Image
# derived values
xcen = ncells/2.0; ycen = xcen

# Make beam grid
beamGrid = FArray.FArray("Beam", [ncells,ncells])
cen = ncells/2
for ix in range(0,ncells):
    x = float (ix -xcen)
    for iy in range(0,ncells):
        y = float (iy -ycen)
        dis = math.sqrt(x*x+y*y)
        val = BeamShape(dis,fi,gau,flux,FWHM,err)
        beamGrid.set(val,ix,iy)
# End loop over image

# Subimage
blc = [ncells/2-nsave/2,ncells/2-nsave/2]
trc = [ncells/2+nsave/2,ncells/2+nsave/2]
beam = FArray.PSubArr (beamGrid, blc, trc, err)
```

```
# Create descriptor
desc=ImageDesc.ImageDesc("Descriptor")
dict = desc.Dict
dict["object"]="PARBeam"
dict["teles"]="Model"
dict["instrume"]="Model"
dict["bitpix"]=-32  # Floats
pos = [0,0]
dict["minval"]=FArray.PMax(beamGrid,pos)
dict["maxval"]=FArray.PMin(beamGrid,pos)
dict["naxis"]=2
dict["ctype"]=["RA---SIN","DEC--SIN","FREQ    ","    ","    ","    ","    "]
dict["inaxes"]=[nsave,nsave,1,1,1,1,1]
dict["cdelt"]= [-cells/3600.0, cells/3600.0, 1.0, 1.0, 1.0, 1.0, 1.0]
dict["crval"]= [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
xcen = nsave/2; ycen = xcen
dict["crpix"]= [xcen+1.0, ycen+1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
desc.Dict = dict

# write to FITS
imag = Image.PFArray2FITS(beam, beamFile, err, outDisk=outDisk, oDesc=desc)
OErr.printErrMsg(err, "Error Writing Beam")

# Shutdown Obit
OErr.printErr(err)
del ObitSys
```

G. OTFSCal documentation

Following is the documentation for OTFSCal. Most of the parameters have sensible defaults and need not be changed, only change the ones necessary.

OTFSCal Imaging/selfcal task for radio OTF data
Type: Task
Use: Processing of radio single dish imaging data allowing "self-calibration" of atmospheric/instrumental effects

The following operations are performed:

1) Selected data are copied to outOTF applying and calibration and selection.
2) (optional) common mode calibration wrt a prior model.
3) (optional) common mode calibration assuming flat (zero) sky
4) Self calibration cycle: The data is imaged applying the current calibration, deconvolve and a common mode calibration is made wrt the CLEAN model for time scales longer than the solution interval for that cycle (solnInt*solnMult[i]). The solution type, determined by doOffset, is either "Common" (common mode only) or "Both" (common mode and detector offsets) Details are given below.
5) (optional) Data are flagged by comparison with last CLEAN model.
6) Final imaging using final calibration and any editing.

Parameters:

DataType

Only 'FITS' is supported

inOTF.

FITS input OTFdata

inDisk

Input OTF data file disk drive #

PSFFile

FITS file containing instrumental PSF

PSFDisk

Disk # for PSF file

Data selection

Targets

List of targets (pointings) to be include. blank is all.

Scans

Beginning and ending scan numbers to include 0 => all

Feeds

List of detector numbers to include, 0=> all

timeRange

Time range of the data to be included, Start time (days), end time relative to ref. date Use dhms2day to convert from human readable form

BChan

First channel number to image, 0=>1. Channel numbers are 1 relative as defined in the input data file.

EChan

Highest channel number to to include in image, 0 => max

RChan

Channel number to restart CLEAN 0=> BChan

chInc

Increment between channels to image in spectral cube.

chAvg

Number of channels to average, 0=> all

doCalib

If true, apply Cal or Soln table, Cal used if exists

keepCal

.f True, include "cal-on" data in image. 'For CCB data, use FALSE

gainUse

Cal/Soln table version number to apply. 0=> highest.

flagVer

FG table to use for editing. 0 => highest.

Output files

outDType

'FITS' or 'AIPS' type of image output Defaults to 'FITS'.

outOTF

Output OTF data file, any existing file deleted

outOTFdsk

Output OTF disk number

outFile

Ending of output FITS image file name
Dirty image file name will end in "Dirty.fits"
Clean image file name will end in "Clean.fits"
Image output weight file name will end in "Wt.fits"

outName

Ending of output AIPS Image Name, Name = source_name+Stokes+outName

outClass

Output image class. Default = 'Map'

out2Class

Output gridding class. Default = 'Wt'

outSeq

Output image sequence number.

outDisk

The disk drive # of output images. 0 => highest with space (note: map and Beam go on same disk. with space. default = outDisk

The following control imaging:

This task images selected single dish calibrated data in Obit OTF format (FITS only). The data are convolved onto the specified grid and "CLEANed" using a data-based method. A Hogbom CLEAN is performed until the peak abs. residual drops below minFlux or the number of iterations reaches Niter.

RACenter

Center of the field in RA to image (deg) If RACenter and DecCenter are both 0, the position of the first entry in Targets is substituted. Note: the image should be defined with sufficient regions around the edge for the convolution function. Pixels with less that minWt convolution weight will be magic value blanked in the output.

DecCenter

Center of the field in Dec to image (deg)

xCells

Cell spacing (asec) in RA **yCells**

Cell spacing (asec) in Dec
(xCells would be a good idea)

nx

Number of cells in RA

ny

Number of cells in Dec

clipData

Data values with abs value greater than clipData are flagged (not included) in the imaging

minWt

Minimum sum of convolution weights as a fraction of the maximum. This is used to exclude regions at the edge of the region covered which are poorly sampled.

Proj

Projection string "-SIN", "-ARC", "-TAN", def "-SIN"

ConvType

Convolving function type: def 5
0 = pillbox, 2 = Sinc, 3 = Gaussian, 4 = Exp*Sinc, 5 = Spheroidal wave

ConvParm

Convolving function parameters:
Pillbox,: no parameters
Sinc,
ConvParm[0] = half-width in cells,def 3.0
ConvParm[1] = Expansion factor def 1.55
Gaussian,
ConvParm[0] = half-width in cells, def 3.0
ConvParm[1] = Gaussian with as fraction or raw beam,def 1.0
Exp*Sinc
ConvParm[0] = half-width in cells, def 2.0
ConvParm[1] = 1/sinc factor (cells) def 1.55
ConvParm[2] = 1/exp factor (cells) def 2.52
ConvParm[3] = exp power def 2.0
Spheroidal wave
ConvParm[0] = half-width in cells def 3.0
ConvParm[1] = Alpha,def 5.0
ConvParm[2] = Expansion factor (not used)

deMode

If True, then subtract the mode of the pixel distribution when forming the dirty image. The intent is to remove any residual image wide calibration biases.

deBias

If True, subtract calibration bias from image when forming the dirty image. This consists of imaging the calibration corrections with a much wider gridding function and then subtracting the image. The intent is that it corrects for spatially variant calibration biases. NB: this doesn't have quite the intended effect

doScale

If True, then scale image by ratio of beam areas determined from convolving the Gridding function with the assumed telescope PSF.

doFilter

If True, filter out of band noise, i.e. spatial frequencies that correspond to power outsize the physical aperture of the instrument. This should generally be left on.

The following control CLEANing:

CLEAN boxes can be specified by either CLEANbox, autoWindow or interactively using dispURL.

CCVer

Output CC table version number, 0=> add new

CLEANBox

A 4x50 array with the specification of a search area. BOX(1,i)=-1 indicates a circle of radius BOX(2,i) pixels centered on (BOX(3,i), BOX(4,i))
BOX(1,i) >= 0 indicates a rectangular box. 0 => full and inner fields. Note: the default box is the entire image

autoWindow

If true, automatically set boxes around significant emission.

Gain

The CLEAN loop gain. 0 => 0.10.

minFlux

Stop Clean when abs(resid. image max) < Flux (Jy)

Patch

Minimum half width of the portion of the beam which is used in the minor CLEAN. Default all

Niter

CLEAN iteration limit. If 0, the dirty image is produced.

BeamSize

CLEAN restoring beam size in asec. If zero, Use estimated instrumental resolution

maxPixel

The maximum number of pixels that are searched for components in each major cycle. < 3000 => 20050. This number affects the CPU usage significantly. Too many causes the task to search over many points it will never use. Too few causes the task to do many more small major cycles, also at great expense. Use this with great caution, but big wins are possible using larger sizes on very large Cleans.

noResid

If True, do not include residuals in resultant images.

doRestore

Restore CLEAN components to images?

The following control the initial and self-calibration. An initial common model calibration using either a flat (zero) or given prior can be done followed by an iterative imaging/deconvolution/recalibration scheme. In each cycle, the previous best estimate of the sky model (CLEAN model) is used as the prior which is used to generate a model time-stream for each detector. The model time-streams are subtracted from the observed time-streams to generate a set of residual time-streams. These residual time-streams are fitted to derive a new set of estimates of the background signals to be subtracted. In general, the timescale should be reduced in this sequence of iterations. If doOffset=True:
In each scan or offsetFact*solution interval, whichever is shorter, the median residual for each detector is taken as its offset and the median of the remaining residuals for all detectors in each integration is fitted with piecewise linear

terms of length the solution interval. If doOffset=False:
In each scan, the median of the residuals for all detectors in each integration is fitted with piecewise linear terms of length the solution interval.

priorFile

If not "None" the FITS file name of the image to be used as the prior for the calibration. This model is used to compute model and residual time-streams and of which the high pass time-filtered median residual ("common mode") is entered into the calibration tables to be subtracted from all subsequent processing.

priorDisk

Disk number of prior FITS file; 0=cwd

priorMod

If True, use the CC table as the prior model, else, the pixels.

priorInt

Shortest timescale (sec) to pass in the residual filtering

commonInt

If > 0.0 then do a common model initial calibration similar to that with the prior model except using a flat, zero sky as the prior.

solnInt

Minimum timescale for low-pass filtering of residual data. This is actually the time increment in the OTFCal table. Actual solution intervals should be no less than this and twice this value is fairly conservative.

solnMult

For each non zero entry a pass of self calibration is done using a shortest timescale ("solution interval") of residual filtering of solInt*solnMult[i]. A value less than 1.0 terminates the sequence. This should probably be a monotonically decreasing series such as (5.0,3.0,2.0).

minResFlx

Pixel values below minResFlx in the model used for each residual calibration are set to this value.

maxResFlx

Pixel values above maxResFlx in the model used for each residual calibration are set to this value.

doOffset

If True then in each cycle of self-calibration detector offsets are determined from the median residuals for times offsetFact times longer than the solution interval in addition to the common mode fitting.

offsetFact

Multiplicative factor times current solution interval for the offset solution timescale. NOTE: This should always be significantly larger than 1.

The following control automatic editing:

doEdit

If True apply automated editing. In each flagInt, each detector time-stream is compared with the model time-stream and data with an RMS in excess of max(maxRMS, maxRatio*model_RMS) is flagged

flagver

Flag table on outOTF to use

flagInt

Flagging interval (sec)

maxRMS

Maximum allowable detector residual RMS in Jy

maxRatio

Max. allowable ratio to equivalent model RMS

prtLv

Print level, 0=>none

Interactive display

dispURL

The URL of the display server to use. "None"=>none "ObitView" = "http://localhost:8765/RPC2" This will display image being CLEANed and allow interactive editing of the CLEAN window.

nThreads

If The Obit libraries are compiled with multiple thread operation enabled, this parameter sets the number of threads that can be used for parallel operations. NB: This only improves performance if there are multiple processors and/or cores in the host.

REFERENCES

[1] W. D. Cotton, "Imaging Data From the MUSTANG Bolometer Array," *Obit Development Memo Series*, vol. 4, pp. 1–10, 2008.

[2] W. D. Cotton, "Obit: A Development Environment for Astronomical Algorithms," *PASP*, vol. 120, pp. 439–448, 2008.

[3] W. D. Cotton, ""Mustang Obit User Documentation"," in *ftp://ftp.cv.nrao.edu/NRAO-staff/bcotton/Obit/MustangObit.pdf*, 2008, pp. 1–10.

[4] B. Mason, S. Dicker, P. Korngut, D. Benford, J. Chervenak, M. Devlin, E. Figueroa, J. Forgione, K. Irwin, M. Mello, S. Maher, H. Moseley, R. Norrod, J. Staghun, and S. White, "First Light with MUSTANG: A 90 GHz Bolometer Array for the Green Bank Telescope," in *Frontiers of Astrophysics: A Celebration of NRAO's 50th Anniversary*, ser. Astronomical Society of the Pacific Conference Series, A. H. Bridle, J. J. Condon, and G. C. Hunt, Eds., vol. 395, Aug. 2008, pp. 374–+.

[5] W. D. Cotton, "ObitTalk User Documentation," in *ftp://ftp.cv.nrao.edu/NRAO-staff/bcotton/Obit/ObitTalk.pdf*, 2009, pp. 1–114.