Multisource Peeling in Obit

W. D. Cotton, July 7, 2021

***Abstract*—"Peeling" is the generic name for determining and applying a direction dependent gain to a limited portion of an image, usually to reduce the artifacts due to a particularly strong source. Various artifacts increase in severity with increasing distance from the pointing center and the most troublesome sources may be ones that appear weak due to strong attenuation by the antenna pattern. While a single strong source can frequently be "fixed" using self calibration some variation of direction dependent calibration may be needed if there are several strong and well separated sources. This memo describes a peeling technique for multiple sources with imaging artifacts. Example usage is given.**

***Index Terms*—imaging, interferometry**

I. Introduction

PEELING is a generic technique for applying direction dependent calibration to a portion of a field of view for images derived from radio interferometer arrays. This is usually to reduce the artifacts from a particularly strong source with direction dependent gain effects due to pointing errors, the rotation of an asymmetric antenna pattern or spatial variations in atmospheric phase. These effects produce apparent time varying changes in source brightness or position resulting in non convolutional errors in the image. This technique applied to a single strong source has been covered in [1].

These errors produce artifacts degrading the image. If the direction specific gain corrections can be independently determined for multiple sources and applied to the data, the level of artifacts can be reduced or eliminated. This memo evaluates such a technique using the Obit package [2] [1]. This is an updated version of [1].

II. Peeling

The term "peeling" is used by a number of authors to mean slightly different things although usually some variant of determining and correcting the direction independent gains over some subset(s) of an image. In the following, "peeling" will be used for the process of:

1) Initial image with direction independent self calibration. This will identify any source(s) needing peeling and produce a model of the sky.
2) Generate a sky model (CC table) excluding the peel region.
3) Subtract the Fourier transform of the sky model without the peel source from the initial self calibrated uv data.
4) Self calibrate the peel source only dataset producing a corrected image and gain table describing the differences between the direction independent gains and those for the peel source.
5) Subtract a corrupted version of the peel sky model from the data. This step is composed of
   a) Generate a direction independent self calibrated version of the full data set ("data").
   b) Generate a model visibility data set ("model") identical in structure to "data".
   c) Invert (1/gain) the total peel self cal gain table copying to the "model" data and replace blanked solutions with (1,0). Replacing failed solutions with (1,0) helps deal with the baselines and frequencies for which the peel source is too faint to give valid solutions. When the peel source is too weak, using a gain of (1,0) causes little harm while flagging these otherwise usable visibilities could seriously degrade the final result.
   d) Applying the inverse corruption table to the Fourier transform of the peeled sky model writing to the "Model" dataset.
   e) Subtract the "Model" data a visibility at a time from "data".
6) Repeat steps 2-5 per source to be peeled with the initial CC table and uvdata file being the output of the previous cycle.
7) Image the final "data" dataset which should result in an image with greatly reduced artifacts.
8) Restore the corrected peel sky model to the final CLEAN image.

A. "Manual" Peeling in Obit

The basic components of the process outlined in Section II, excluding the first and the last 2 steps, are implemented in python module python/PeelScripts.py.

1) SelectCC: The step in Section II bullet 2 is implemented in routine SelectCC. This generates a CC table excluding the region around position pos radius in extent. The documentation for routine SelectCC is given in Figure 1.

2) UVSub4Peel: The step in Section II bullet 3 is implemented in routine UVSub4Peel. This sets up for the subtraction of the sky model missing the peel source which generates a new uv dataset. The routine returns a UVSub task interface object which may be further modified prior to execution. The documentation for routine UVSub4Peel is given in Figure 2.

3) ImagePeel: The step in Section II bullet 4 is implemented in routine ImagePeel. This sets up imaging with self calibration of the peel source only data set. A task object for MFImage is returned which may be further modified prior to execution. The documentation for routine ImagePeel is given in Figure 3.

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

[1] http://www.cv.nrao.edu/∼bcotton/Obit.html

4) SubPeel: The step in Section II bullet 5 is implemented in routine SubPeel. This routine subtracts the corrupted sky model of the peel source from a copy of the original data. Imaging this dataset will give the full field of view without the peel source. The documentation for routine SubPeel is given in Figure 4.

5) RestorePeel: The CLEAN model of the peeled image can be added back into CLEAN image derived from the data produced in the previous step. Images derived by MFImage will have all total intensity planes restored. The documentation for routine RestorePeel is given in Figure 5.

III. Example Multisource Peel

The test dataset is from MeerKAT observations at L band This field contains multiple strong sources with varying direction dependent gains giving rise to multiple sets of artifacts. Initial imaging used Obit/MFImage with a field of view of 1.2° with outliers to 1.8° and a Briggs Robust factor of -1.5. Cleaning used up to 100,000 CLEAN components and 2 iterations of phase only self calibration. Four strong sources with radial patterns of artifacts are seen in Figure 6. The artifacts are from some combination of antenna pointing errors, asymmetries in the antenna pattern and ionospheric phase variations.

The troublesome sources were peeled as described in Section II and the results shown in Figure 7. Detailed before and after images of the regions around the peeled sources are shown in Figure 8. The artifacts from the direction dependent gains are almost completely removed.

A. Scripting

The probability of correctly getting through a complex sequence like that needed here manually is vanishingly small. A template work file that can be suitable modified and used to "cut and paste" commands into python is shown in Figures 9&10 and is included in the Obit distribution as $OBIT/python/PeelWork.py (svn 633 and later). This starts with a self calibrated dataset and the derived image with a CC table giving the self calibrated sky model. This "script" liberally generates new versions of datasets and some cleanup may be needed during the processing if disk space becomes an issue.

IV. Discussion

A relatively generic technique for peeling sources showing artifacts is described and an example of its application are shown. Clear artifacts resulting from the varying antenna gain/atmospheric phase towards multiple sources are almost completely eliminated. It does not appear to be necessary to reimage the data after subtracting each source when the sources to be peeled are of comparable strength. Residual artifacts are at a higher level for more resolved and complex sources peeled.

This procedure makes heavy use of DFT model visibility calculations. A local built from source installation of Obit allowing usage of a GPU can dramatically speed the process.

Acknowledgment

I would like to thank the MeerKAT staff, especially Fernando Camilo for assistance and for providing the MeerKAT data.

References

[1] W. D. Cotton, "Manual Peeling in Obit," *Obit Development Memo Series*, vol. 54, pp. 1–5, 2017. [Online]. Available: ftp://ftp.cv.nrao.edu/NRAO-staff/bcotton/Obit/ManualPeel.pdf

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

```
SelectCC(im, inCC, outCC, radius, peelPos, err)
    Select/copy CCs more than radius from peelPos

    This generates a CC table which can be subtracted from the
    uv data and remove all sources but the peel source area.
    * im     = Python Image with CC Tables
    * inCC   = input CC version
    * outCC  = output CC version
    * radius = radius (deg) of zone of exclusion
    * peelPos= [RA, Dec] in deg.
    * err    = Python Obit Error/message stack
```

Fig. 1. SelectCC function

```
    UVSub4Peel(uv, source, im, inCC, err, nfield=1, doGPU=False,\
        gainUse=1, flagVer=-1, nThreads=1, noScrat=[0, 0, 0], \
        taskLog='', debug=False)
        Sets up for subtraction of non peel sources from data

        UV data should be have calibration tables from self calibration
        Output data will be on the same disk as the input, seq=1,
        class='4Peel' and with the name of the source (up to 12 char)
        Returns UVSub task object
        * uv        Dataset to be subtracted from
        * source    source name
        * im        Python Image with CC Tables
        * inCC      input CC version, should have had the peel source
                    CCs removed using SelectCC
        * err       Python Obit Error/message stack
        * doGPU     Use GPU if available?
        * nfield    Number of facet images
        * gainUse   CL (SN) table to apply, -1=> no cal
        * flagVer   FG table to apply, -1=> no flag
        * noThreads number of threads to use
        * noScrat   AIPS disks not to use for scratch
        * taskLog   Log file
        * debug     If True leave debug Input file in /tmp
```

Fig. 2. UVSub4Peel function

```
ImagePeel(uvsub, peelPos, err, nxy=512, Niter=1000, minFlux=0.001, \
    maxPSCLoop=2, minFluxPSC=0.01, solPInt=1.0, minSNR=3.5, \
    Robust=0.0, doGPU=False, seq=1, nThreads=1, noScrat=[0, 0, 0], \
    taskLog='', debug=False)
    Sets up to image subtracted uv data from UVSub4Peel,
    self calibrate peel source.

    Only does A&P self cal
    Returns MFImage task object, output image "IPlMod', uv 'UVPeel', Seq seq
    * uvsub     task object from  UVSub4Peel
    * peelPos   [RA, Dec] in deg of source to peel
    * err       Python Obit Error/message stack
    * nxy       Size in pixels of x,y
    * Niter     max number of iterations
    * minFlux   Min flux density first CLEAN
    * maxPSCLoop max number A&P self cal loops
    * minFluxPSC min peak for self cal
    * solPInt    solution interval for self cal
    * minSNR     min SNR of self cal solutions
    * Robust     Briggs Robust factor
    * seq        Sequence number for output
    * doGPU      Use GPU if available?
    * nThreads  number of threads to use
    * noScrat   AIPS disks not to use for scratch
    * taskLog   Log file
    * debug     If True leave debug Input file in /tmp
```

Fig. 3. ImagePeel function

```
SubPeel(uv, source, imp, uvp, err, flagVer=0, nThreads=1, \
    addBack=False, seq=999, doGPU=False, noScrat=[0, 0, 0],
    taskLog='', debug=False)
    Subtract Peel model w/ solutions, then optionally
    add back w/o corruptions

    UV data should have calibration tables from self calibration
    Output data will be on the same disk as the input, seq=seq,
    class='PelSub' and with name = source (up to 12 char).
    Returns Peel source subtractedd data
    * uv        Dataset with cal tables
                Needs at least the self cal gain table
    * source    source name
    * imp       Peel source model (CC table from ImagePeel)
    * uvp       UV data the result of peel (ImagePeel)
    * err       Python Obit Error/message stack
    * seq       Sequence number for output
    * addBack   Add model back to data w/o corruptions?
    * flagVer   FG table to apply, -1=> no flag
    * nThreads  number of threads to use
    * doGPU     Use GPU if available?
    * noScrat   AIPS disks not to use for scratch
    * taskLog   Log file
    * debug     If True leave debug Input file in /tmp
```

Fig. 4. SubPeel function

```
RestorePeel(peelMod, CCVer, image, err)
    Restore CCs from one image onto another

    If images are ImageMF then multiple planes restored.
    * peelMod   Image with CC table (as Image)
    * CCver     CC version on peelMod to restore
    * image     Output Image to which components to be added
    * err       Python Obit Error/message stack
    * nThreads  number of threads to use
```

Fig. 5. RestorePeel function

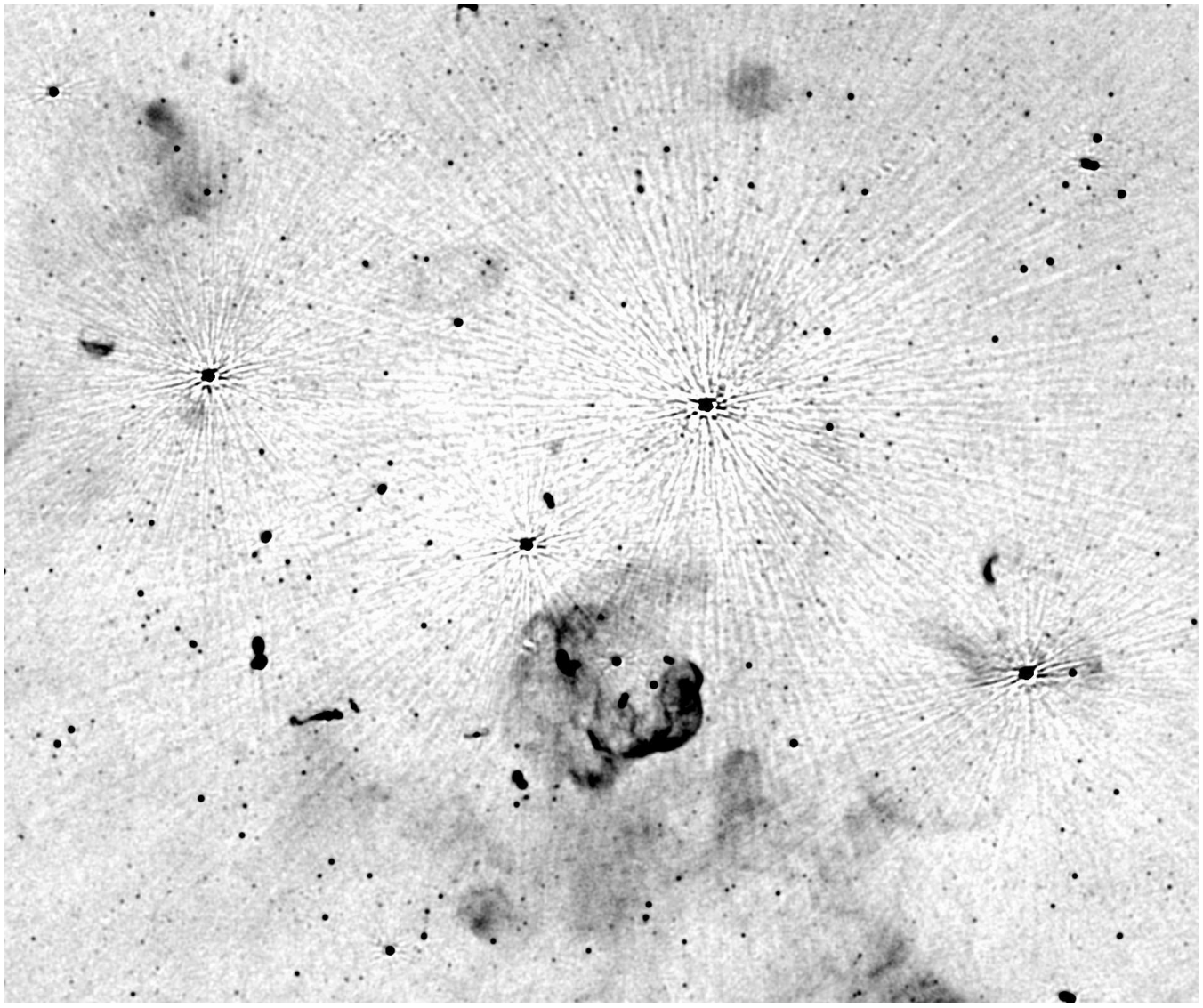

Fig. 6. Negative grayscale of portion of a field observed with MeerKAT without peeling. The stretch is -30 to 200 μJy/beam and the field shown is 55' $\times$ 46'.

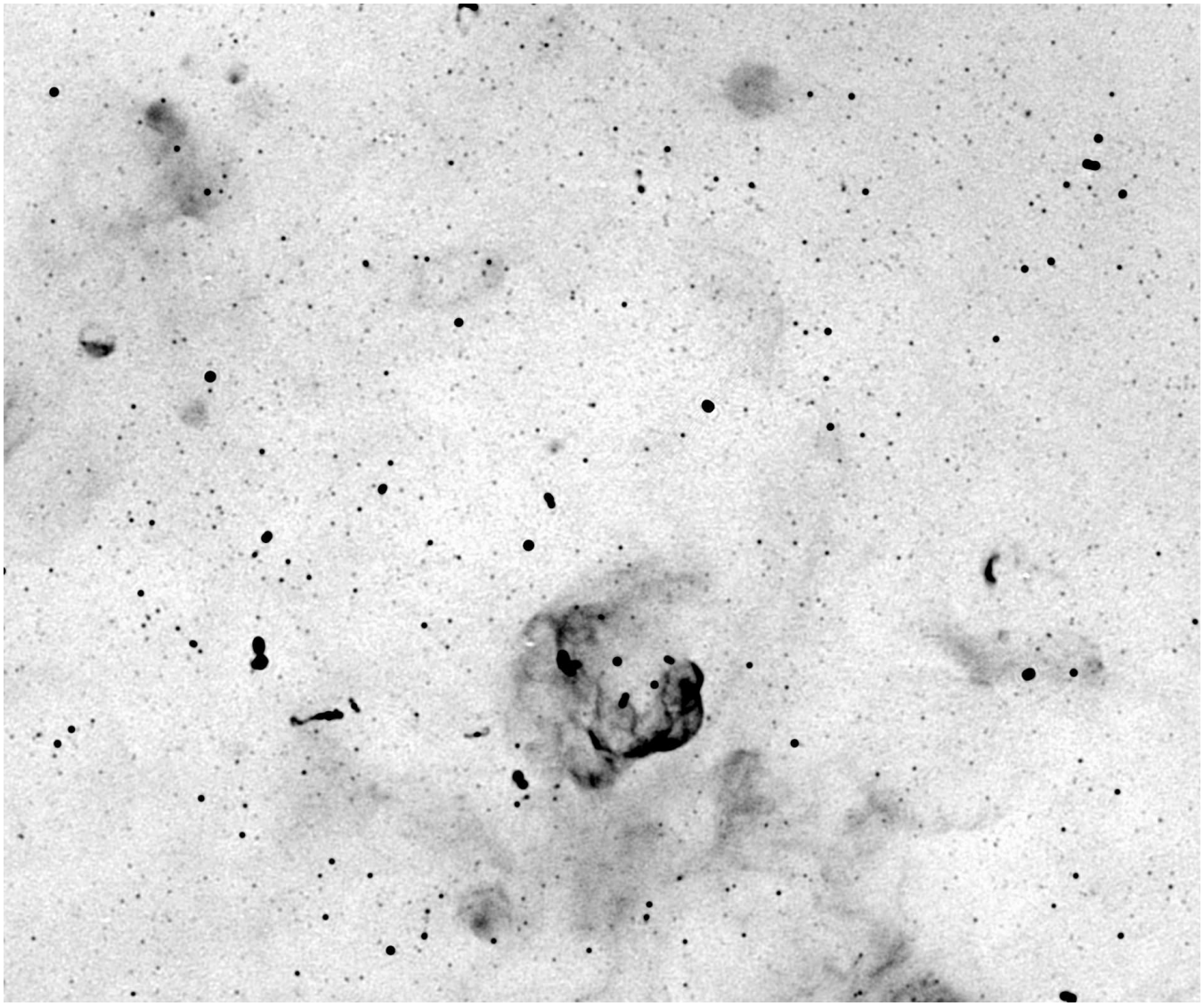

Fig. 7. Like Figure 6 but with peeling.

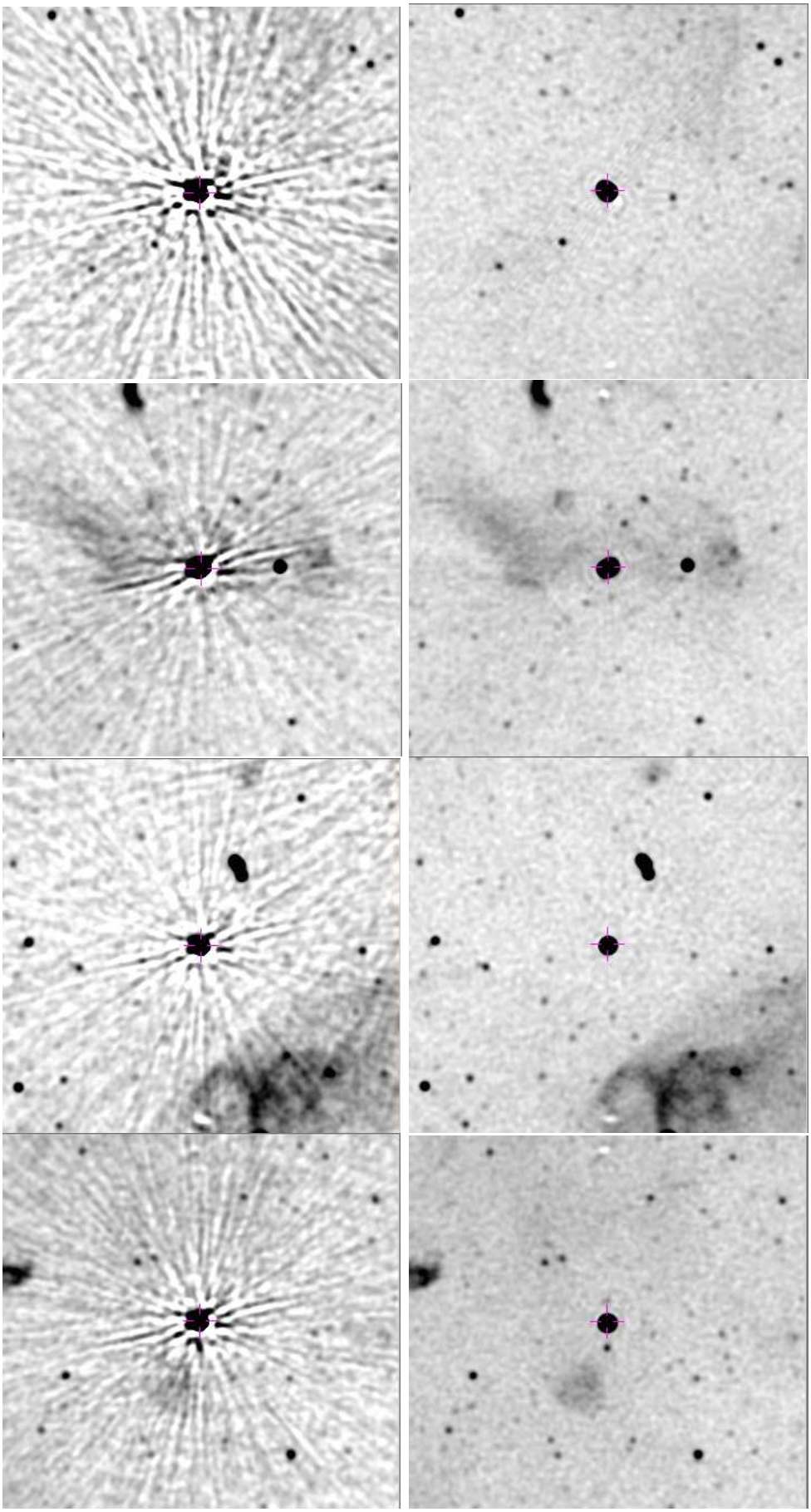

Fig. 8. Negative grayscale detailed comparison of before (left) and after peeling (right) for the 4 sources peeled. The stretch is -30 to 200 μJy/beam and the field shown is 7.8' $\times$ 7.3'. Crosses show the location of the peeled source.

```
#---------------------mySource -------------------------------------
source = 'mySource'; gainuse=0;
AUname=source; AUclass='SCal'; AUdisk=2; AUseq=1   # Initial uv data
AIname=source; AIclass='IPScal'; AIdisk=1; AIseq=1 # Self cal image w/ CC table
# First source to peel
ra='01:02:18.71';dec='-75:46:52.6'; seq=1

# Second source to peel
ra='01:11:33.96';dec='-75:38:10.5'; seq=2; gainuse=-1
AUclass='UPeel'; AUdisk=2; AUseq=seq

# Third source to peel
ra='01:13:21.34';dec='-75:28:21.2'; seq=3; gainuse=-1; AUseq=seq

# parameters likely to be changed
oDisk = 1       # output disk
FOV   = 1.2     # Radius of full field of view (deg)
Robust=-1.5     # Briggs robust factor
Niter = 100000  # Maximum number of CLEAN components
minFlux=50.0e-6 # CLEAN depth (Jy)
nthreads=24     # How any threads?
doGPU = False   # Use GPU for model calculation

from PeelScripts import *

# For each pool source do steps 1-4
# May need to clean up uv data files after they are no longer needed

# 1) select other non peel CCs
x=Image.newPAImage('inIm',AIname, AIclass, AIdisk, AIseq, True,err)
rad = ImageDesc.PHMS2RA(ra); decd= ImageDesc.PDMS2Dec(dec)
peelPos = [rad,decd]
# select within 60" of peelPos
SelectCC(x,seq,seq+1,60./3600,peelPos,err)

# 2) Subtract others
uvi=UV.newPAUV('inUV',AUname, AUclass, AUdisk, max(1,AUseq-1), True,err)
uvs=UVSub4Peel(uvi,source,x,seq+1,err,nThreads=nthreads,gainUse=gainuse)
uvs.doGPU=True; uvs.outSeq=seq
uvs.g

# 3) Image peel source
us=UV.newPAUV('inUV',AUname, '4Peel', AUdisk, AUseq, True,err)
mm=ImagePeel(uvs,peelPos,err,nThreads=nthreads,maxPSCLoop=4,solPInt=0.5, \
             minFlux=0.0001,seq=seq)
mm.dispURL= "http://localhost:8765/RPC2"
mm.Sources=[source]; mm.Niter=5000; mm.refAnt=59; mm.minFluxPSC=0.005; mm.minFluxASC=0.005
mm.Robust=-1.5; mm.autoWin=False; mm.CLEANBox=[-1,5,0,0]
maxAWLoop = 1
addParam(mm,"maxAWLoop", paramVal=maxAWLoop, shortHelp="Max. middle CLEAN loop", \
        longHelp="  maxAWLoop....Max. middle CLEAN Loop count\n"+ \
        "              Override the default behavior for the autoWin middle loop\n"+
        "              if > 0.\n")
mm.doGPU=doGPU
mm.g
```

Fig. 9. Example Peeling work file

```
# 4) remove peel source
if (seq==1):
    uv = UV.newPAUV('inData', AUname, AUclass, AUdisk, seq, True,err)
else:
    uv = UV.newPAUV('inData', AUname, AUclass, AUdisk, seq-1, True,err)

uvp=UV.newPAUV('inUV', source[0:12], mm.out2Class, mm.out2Disk, mm.out2Seq, True,err)
imp=Image.newPAImage('inIm', source, mm.outClass, mm.out2Disk, mm.outSeq, True,err)
uvpeel=SubPeel(uv,source, imp, uvp, err, doGPU=doGPU, seq=seq, addBack=False,  nThreads=nthrea

# Image data with no further selfcalibration, may need to fiddle
mf=ObitTask('MFImage')
setname(uvpeel,mf);mf.Sources = [source];
mf.doCalib=-1;mf.gainUse=0; mf.flagVer=1; mf.OutlierDist=1.2*FOV; mf.OutlierFlux=0.001
mf.Stokes='I';mf.doBand=-1; mf.BPVer=1; mf.doPol=False; mf.PDVer=2;
mf.outClass='IPeel';mf.outSeq=1; mf.FOV=FOV; mf.OutlierSize=515; mf.minPatch=500
mf.Niter=Niter; mf.BLFact=1.01; mf.PBCor=False; mf.prtLv=2; mf.autoWindow=True
mf.outDisk=oDisk; mf.out2Disk=oDisk; mf.out2Seq=1; mf.Catalog='AllSkyVZ.FIT'
mf.doGPU=doGPU; mf.nThreads=nthreads; mf.Robust=Robust; mf.maxPixel=1000000
mf.Gain=0.05;mf.autoWindow=True; mf.ccfLim=0.50; mf.minFlux=minFlux;
mf.maxPSCLoop=0; mf.minFluxPSC=0.01; mf.solPMode='P'; mf.solPType='L1';  mf.solPInt=0.50;
mf.maxASCLoop=0; mf.minFluxASC=1.0; mf.solAMode='A&P'; mf.solAInt=10.0; mf.solAType='L1'
mf.avgPol=True; mf.autoCen=1000.0; mf.noNeg=False;
mf.dispURL= "http://localhost:8765/RPC2"
mf.logFile=source+'.MFImage.log'
maxAWLoop = 1
addParam(mf,"maxAWLoop", paramVal=maxAWLoop, shortHelp="Max. middle CLEAN loop", \
        longHelp="  maxAWLoop....Max. middle CLEAN Loop count\n"+ \
        "               Override the default behavior for the autoWin middle loop\n"+
        "               if > 0.\n")
minFList = [0.00007,0.00005,0.00005, 0.00005]
addParam(mf,"minFList", paramVal=minFList, \
        shortHelp="minFlux list after SC", \
        longHelp="  minFList....minFluxes to use in IPol after selfcals\n")

mf.g  # run imaging

# Restore peeled sources
xpeel = Image.newPAImage('peel',source[0:12],'IPeel', mf.outDisk, mf.outSeq,True,err)
nseq = seq; CCVer=1
for iSeq in range(1,nseq+1):
    xi = Image.newPAImage('mod',source[0:12], mm.outClass, mm.outDisk, iSeq,True,err)
    RestorePeel(xi, CCVer, xpeel, err)
```

Fig. 10. Example Peeling work file continued