EVLA Memo 67

W PROJECTION: A NEW ALGORITHM FOR NON-COPLANAR BASELINES

T.J. CORNWELL, K. GOLAP, AND S. BHATNAGAR

Abstract. Wide-field imaging is key to many of the science goals of the EVLA in both Phase I and Phase II. Optimization of wide-field imaging algorithms is therefore vital. In this memo, we consider the existing algorithms for dealing with non-coplanar baselines and propose a new algorithm with markedly superior performance. At roughly equivalent levels of accuracy, this new algorithm, which we call "w-projection", is about an order of magnitude faster than the corresponding facet-based algorithms. The new algorithm is now available in AIPS++.

1. Overview of the non-coplanar baselines problem

The response of a narrow-band phase-tracking interferometer to spatially incoherent radiation from the far field can be expressed by the following relation between the spatial coherence, or 'visibility', $V(u,v,w)$, and the spectral intensity, or brightness, $I(\ell,m)$;

$$V(u,v,w)=\int\int\frac{I(\ell,m)}{\sqrt{1-\ell^2-m^2}}\;e^{-2\pi i[u\ell+vm+w(\sqrt{1-\ell^2-m^2}-1)]}d\ell dm \tag{1}$$

In this equation, the baseline coordinates, (u,v,w), and direction cosines, (ℓ,m) have their usual definitions.

When the term $2\pi w(\sqrt{1-\ell^2-m^2}-1)$ is much less than unity, it may be ignored, and a two-dimensional Fourer relationship results. When this term is comparable to or exceeds unity, a two-dimensional Fourier transform cannot be used. The value of this term is roughly:

$$\frac{B\lambda}{D^2} \tag{2}$$

where B is the maximum baseline length, D is the antenna diameter, and λ is the observing wavelength. This effect (called "non-coplanar baselines") is therefore most troublesome for the larger VLA configurations observed at longer wavelengths. Also, as Perley and Clark discuss (2003), this factor worsens for small antennas such as will be used in the Large N, Small D design for the SKA.

The non-coplanar baselines effect consists of two related but somewhat distinct parts. First there is a position-dependent phase-shift due to the term $2\pi w(\sqrt{1-\ell^2-m^2}-1)$. Second, when the Fourier plane sampling $S(u,v,w)$ is taken into account, the actual size structures sampled by a given physical pair of antennas varies across the field of view.

Date: 2003 December 16.

2. Review of existing algorithms

A number of algorithms for dealing with non-coplanar baselines have been proposed. In the course of our development of our new algorithm, we reviewed all of those listed below to see if any substantial increase in performance could be made.

2.1. **Fourier sum.** Equation (1) may be numerically integrated using a pixellated image. This can be arbitrarily accurate but is nearly always prohibitively expensive.

2.2. **Component models.** The sky brightness can be modelled by a collection of discrete components, drawn from a fixed repertoire of component types, the Fourier transform of which may be calculated analytically. The *AIPS++* imaging package in fact has two parallel paths for prediction of coherence data - from an image, and from a component model. In the context of wide-field imaging, this approach allows relaxation of some of the tolerances on, for example, number of facets, since the brightest emission is modelled by exactly transformable components.

2.3. **Fourier sum of bright pixels.** There is a simplicity to dealing only with images (rather than images and component models). A useful compromise, therefore, is to perform Fourier sums for the bright pixels and some other more approximate transform for the other pixels. The SDE dragon task did this to good effect - using the image-plane facet approach (see below) for the low brightness emission.

2.4. **Warped snapshots.** In figure (1), we show schematically the projection of a snapshot observation for a planar array onto the celestial sphere. Since the array is planar, the instantaneous PSF has 100% sidelobes in the direction of the zenith. Hence in performing a standard two-dimensional transform, sources on the celestial sphere (A,B,C,D) are projected onto the tangent plane at (A",B",C",D"), instead of to the true locations (A',B',C',D'). As the local zenith moves around on the celestial sphere (*i.e.* as the Earth rotates!), these erroneously projected points move around on the tangent plane. We note that the name "non co-planar baselines" for this effect comes from this description.

Bracewell (1984) noted that for instantaneously planar sampling, w can be eliminated from equation (1). We can show that if Z is the Zenith angle, and χ is the parallactic angle at the time of observation, then the relationship between sky brightness and visibility can then be expressed as:

$$V(u,v,w) = \int\int \frac{I(\ell,m)}{\sqrt{1-\ell^2-m^2}} e^{-2\pi i[u\ell'+vm']} d\ell dm \tag{3}$$

where:

$$\ell' = \ell + \tan(Z)\sin(\chi)\left(\sqrt{1-\ell^2-m^2}-1\right) \tag{4}$$

$$m' = m - \tan(Z)\cos(\chi)\left(\sqrt{1-\ell^2-m^2}-1\right) \tag{5}$$

Thus a two dimensional Fourier transform between sky brightness and visibility holds at any instant but at the cost of a coordinate distortion in the sky plane. This can not be corrected via a coordinate transform in Fourier space and so image plane regridding of each snapshot, either before or after deconvolution, is required. Unfortunately, the required high precision image plane coordinate transform is computationally expensive.

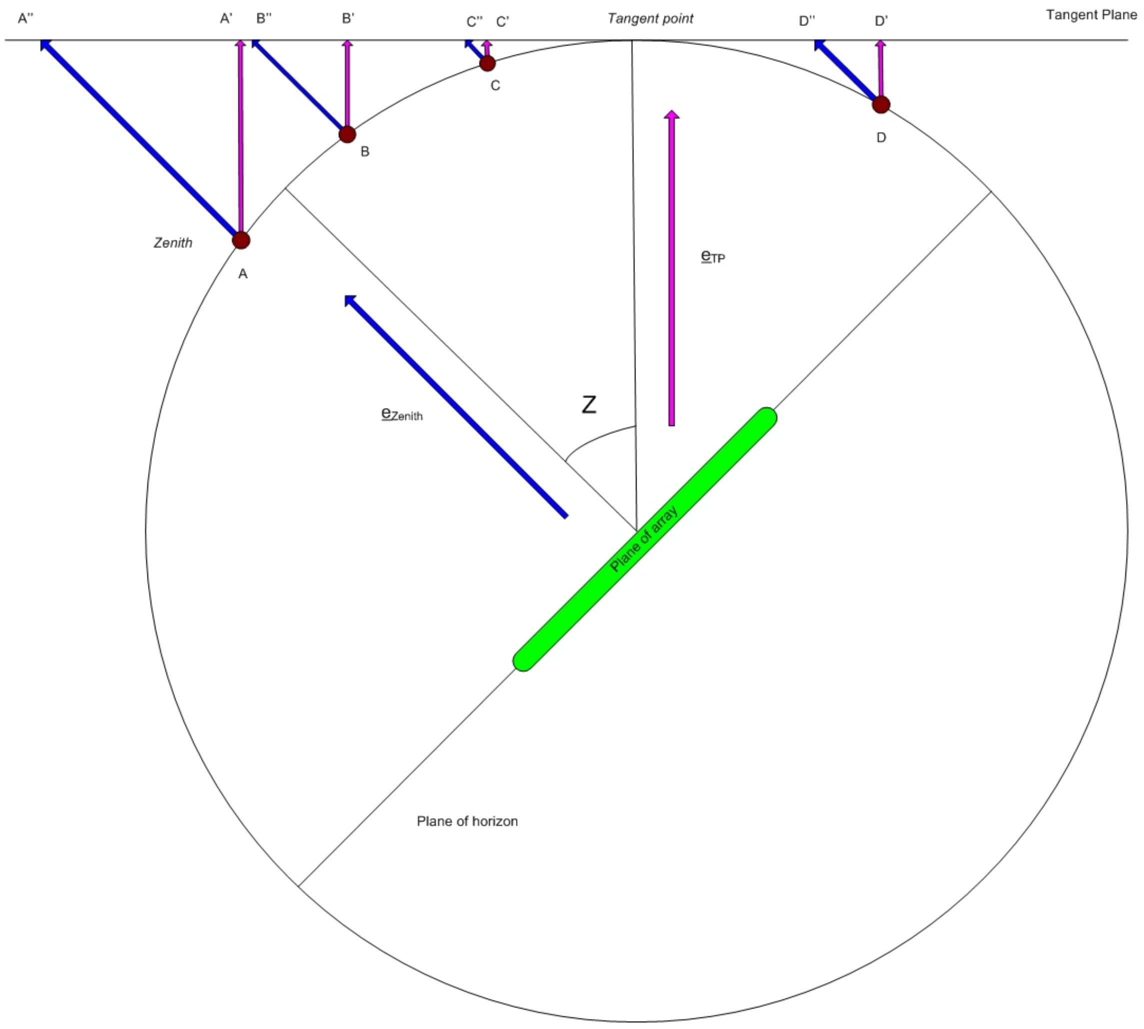


FIGURE 1. Schematic of the projection of a single snapshot onto the celestial sphere. Sources on the celestial sphere at points (A,B,C,D) are projected onto the tangent plane at (A",B",C",D"), instead of to the true locations (A',B',C',D')

It remains a curious and ironic fact that none of the algorithms currently in use (including our new algorithm discussed below) actually make any use of the instantaneous planarity of the array.

2.5. **Image-plane facets.** In this, the most commonly used approach, the image plane is factored into N_{facets} by N_{facets} facets. Each facet is imaged separately in a minor cycle, and then reconciled in a major cycle. The original algorithm demonstrated by Cornwell and Perley (1992) involved making separate images for each facet and then reducing all of these to a common tangent plane after deconvolution. Each facet is naturally deconvolved separately using the appropriate

PSF. The original SDE `dragon` program and the AIPS IMAGR task both use this approach.

The number of facets required along any axis is proportional to the ratio mentioned above:

$$\frac{B\lambda}{D^2} \tag{6}$$

More accurately, the number of facets needed in ℓ and m is:

$$N_{facets} = \frac{\pi\Theta\sigma_W}{\sqrt{32\delta A}} \tag{7}$$

where Θ is the field of view in radians, σ_w is the dispersion in w, and δA is the maximum tolerable amplitude loss. If the positions of the sources away from the phase center are not required to any accuracy, then σ_W should be the residual value after removing a best fitting plane.

As the facet size shrinks all the way down to one pixel, the image-plane facet algorithms becomes just a Fourier sum. One can therefore think of the image-plane facet algorithm as combining the virtues of the direct Fourier sum and FFTs.

2.6. **Fourier-plane facets.** An alternative to image-plane facets is to project the Fourier plane data for each facet onto one tangent plane during imaging (See Sault *et al.*, 1996 for mathematical details). This is fast (involving only a matrix multiply of each (u, v)) and avoids having to deal with a large number of facet images. The *AIPS++* `imager` tool uses this approach.

2.7. **3D transforms.** The non-coplanar problem arises because we are trying to force the use of a two-dimensional Fourier transform (motivated by the efficiency of the FFT). An alternative is to move up to a three dimensional Fourier Transform. Equation (1) may be embedded in a three dimensional space with axes (ℓ, m, n).

$$V(u, v, w) = \int\int \frac{I(\ell, m)\delta\left(n - \sqrt{1-\ell^2-m^2}\right)}{n} e^{-2\pi i[u\ell+vm+wn]} d\ell dm dn \tag{8}$$

This three dimensional Fourier transform may be implemented using an FFT in all axes, or if the range in n is small, FFTs in (ℓ, m) with DFT in n. This approach was implemented in SDE but not often used. The principal drawback is that for large field of view, the interior of the cube is largely devoid of true emission.

2.8. **Some remarks.** It is generally believed that either image- or Fourier-plane faceting is needed to account for the shift-variant nature of the point spread function. In fact, the PSF varies at a level comparable to or below the approximations inherent in assuming decoupling of the facets during the minor cycle.. This means that there is no real advantage to using the local, rather than average, PSF in a major/minor cycle algorithm (Clark 1980). This realization opens up the possibility of using algorithms that calculate one residual image for the entire field. The uv facets algorithm can be viewed as doing this, of course, but there are also other approaches possible. In the next section, we discuss an algorithm that does generate one residual image for the entire field.

3. W reprojection

Since the problem largely originates with the w part of (u, v, w), it is worth asking if there is any way to project w out of the problem. This would then allow a two dimensional Fourier transform to a single image to be used. Frater and Doherty (1980) noted that projection from a single plane w to $w = 0$ is possible, and they proposed using Clean to solve the resulting convolution relationship. However, it is clear that their equation allows reprojection to and from any position in (u, v, w) space from and to the $w = 0$ plane. To derive this result, we must rewrite equation (1) as a convolution between the Fourier transform of the sky brightness and the Fourier transform of an image plane phase term parametrized by w.

$$V(u,v,w) = \int\int \frac{I(\ell,m)}{\sqrt{1-\ell^2-m^2}} G(\ell,m,w)\; e^{-2\pi i[u\ell+vm]} d\ell dm \tag{9}$$

$$G(\ell,m,w) = e^{-2\pi i[w(\sqrt{1-\ell^2-m^2}-1)]} \tag{10}$$

In the Fourier space (after Fourier transformation with respect to (ℓ, m) and with unlimited domain of integration in the image plane), the convolution function due to this term is:

$$G(u,v,w) = \frac{sin\left(\frac{\pi(u^2+v^2)}{2w}\right)}{2w} \tag{11}$$

In practice, this must be further convolved with the convolution function used for gridding (which necessarily damps the divergent behavior at $w = 0$). In general numerical integration is required. The *AIPS++* `qimager` tool uses numerical integration with 8x padding in the image plane, non-linear spacing of w planes, and w-dependent truncation of the aggregate convolution function (at 0.1%).

In figure (2), we show schematically the effect of reprojecting a single (u, v, w) sample on to the $(u, v, w = 0)$ plane. Each sample is spread over the $(u, v, w = 0)$ plane with a footprint roughly proportional to the value of w. This reflects the sensitivity of a single sample point to a range of spatial frequencies across the field of view.

In figure (3), we show the image plane function and Fourier transform for $w = 0, w_{max}/2, w_{max}$ for a typical observation with the VLA. For $w = 0$, the convolution function is simply the usual spheroidal function. For w increasing, the area of the (u, v) plane affected by the projection of any point increases.

Given a model of the sky brightness, we can therefore predict the visibility on the $(u, v, w = 0)$ plane by a two-dimensional Fourier transform in (ℓ, m). Using the convolution function, we may then project this from the $(u, v, w = 0)$ plane to the general point (u, v, w). The calculation in this direction (image to Fourier) is exact (except for numerical errors, of course). Going in the opposite direction proceeds as follows: we project each (u, v, w) point onto the $(u, v, w = 0)$ plane using the w projection function. We then Fourier transform (using a two-dimensional transform) these gridded visibilities to the image plane, where we have thus obtained a dirty (or residual) image on the tangent plane. This residual image may then be used in a minor cycle deconvolution algorithm to update the model.

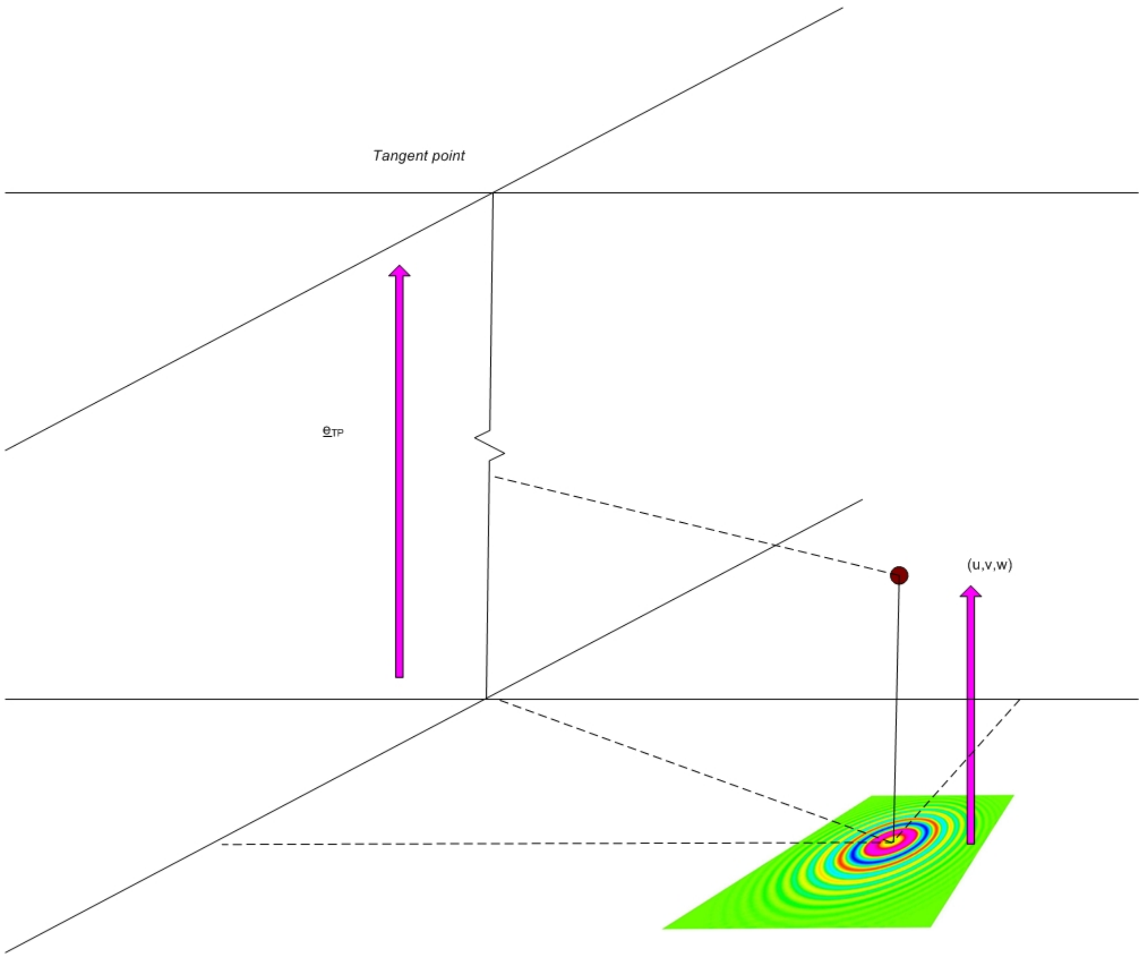


FIGURE 2. Schematic of the projection of a single (u, v, w) sample onto the $(u, v, w = 0)$ plane.

The usual spheroidal gridding function has support 9 by 9 pixels in (u, v). The support of the w-dependent gridding function grows with both w and the field of view, typically up to a largest value of about 70 by 70. We oversample by a factor of 8, and limit the size of the convolution function to 512 by 512 pixels in (u, v). The size of the convolution function per plane is no more than 2 MB. The work involved in gridding with this convolution function increases in direct proportion to the extent in (u, v). The average value of w gridded is close to $w_{max}/2$, so the typical increase in gridding costs is about a factor of 10-20. While a wider convolution function directly leads to more computing, the extra cost incurred by using more planes in w is relatively minor - all that happens is an essentially negligible increase in the memory access time required to get the relevant part of the convolution function. Discrete sampling in w leads to aliases at the scale size of $1/\Delta w$. Placing these aliases outside the field of view requires that w be sampled with similar precision to (u, v). Spacing the w planes evenly in $\sqrt{w}$ reduces this effect to a tolerable level. Beyond this point, the errors are due to the incorrect value of the convolution function being used. Since this error is dependent on the exact value of w with respect to the tabulated values, the resulting error is not strongly correlated with (u, v). Consequently, any errors present should not have very coherent shapes. This is in contrast to the facet approaches where the prediction error is necessarily worse on the longest baselines, and so the resulting error patterns are quite coherent.

We can derive the size of the support region by the following argument. The spread in *e.g.* u is simply a geometric effect: the maximum value of w multiplied by the field of view. Taking the field of view to be λ/D, we find that the number of pixels in the support region along the u axis goes as $B\lambda/D^2$).

$$T_{facets} = N^2_{facets}.N^2_{GCFpoints}.T_{single} \tag{12}$$

$$T_{wproject} = (N^2_{wproject} + N^2_{GCFpoints}).T_{single} \tag{13}$$

where $N_{GCFpoints}$ is the support of the normal gridding convolution function in one axis (typically 7 or 9), and N_{facets} and $N_{wproject}$ are both proportional to $B/(\lambda D^2)$ but with different proportionality constants. We have assumed that the sizes of the normal gridding convolution function and the w projection convolution function add in quadrature. If we take N_{facets} and $N_{wproject}$ to be roughly equal, then for large fields of view, the ratio of these times is roughly the total number of points in the *normal* gridding convolution function. Since the normal gridding convolution function is typically 7 by 7 or 9 by 9, the maximum speedup is between 50 and 100. Allowing for different proportionality constants, we could conservatively expect at least an order of magnitude.

We simulated a 74MHz VLA C-configuration full synthesis on a field at right ascension 13h, declination 45 degrees. The data consisted of 505440 visibility records, each of 8 spectral channels. The sources were generated by taking the ten WENSS sources above 2Jy within 5 degrees of the center. All images were cleaned with 10,000 iterations at a loop gain of 0.1. The uv facets results were generated using the in-memory FTMachine (now the default) which is known to be approximately the same speed as the AIPS IMAGR. The simulation script is given in Appendix A.

The results are shown in Table I. The first column shows the approach used, the second is a robust estimate of the dynamic range after cleaning (peak/(median absolute deviation from the median)), the third is the time in seconds to initialize the convolution functions, and the last column is the time taken to calculate the residual image (involving a transform in both directions). The key results are:

- W projection is about 8 times slower than standard two dimensional Fourier transformation. The standard gridding uses support of 9 by 9 pixels. The average size of the convolution region in w projection is about 30 by 30 pixels so the increased load in gridding for w projection should be about (30*30/9*9) times bigger or a factor of 11. This is reasonably in line with the measured number and certainly indicates that the w projection costs are in proportion to the increase in multiplications in the innermost loop.
- For the same dynamic range, w projection is about an order of magnitude faster than the uv facets approach.

- For w projection, the aliasing performance and initialization time both grow linearly with the number of planes, whereas the residual calculation time is independent of the number of planes.
- For uv facets, the aliasing performance and the residual calculation times both grow linearly with the number of facets (*i.e.* the square of the number per axis).

Method	DR	Initialize (s)	Find residual (s)
Standard FT	35	-	18
uv facets (5 by 5)	2000	-	492
uv facets (7 by 7)	2500	-	565
uv facets (13 by 13)	4700	-	1697
w projection (8)	1600	2	153
w projection (16)	3300	4	159
w projection (32)	6700	9	161
w projection (64)	12000	20	157
w projection (128)	20800	38	158

Table I: Performance of various methods on simulated 74MHz C configuration observations.

In figure (4), we compare the restored images obtained from uv facets (13 by 13) and w projection (128). A few comments:

- The errors in the uv facets image are centered around the bright sources whereas those in w projection image are more isotropic (though some systematic aliasing is still present). The peak error sidelobe in the uv facets image is about 0.3%. In practice, this would be reduced by two effects: first, phase selfcalibration will remove phase errors on the longest baseline on the few brightest sources, and second, the brightest source is often placed on a facet center. Taken together this would probably reduce the peak error in this case to 0.1%, which is almost certainly not noticeable in typical cases where uv facets are used.
- The time differential in residual calculation is about a factor of eight, and the error performance about 4.4, both in favor of w projection. Overall, the uv facets CLEAN took 10675s, and the w projection CLEAN took 1699s, so in aggregrate the speed ratio is about a factor of six. The uv facets algorithm has a relative advantage early in iteration because it need not Fourier transform (to the Fourier plane) any facets empty of emission. This accounts for the difference between the factor of eight for residual calculation and six overall.
- Extrapolating the uv facets behavior, we find that to obtain the same dynamic range as the best w projection image, about 25 by 25 facets would be required, at a cost of about 24 compared to w projection.

4. Discussion

We have demonstrated a new algorithm for correcting the non-coplanar baselines effect. This has superior performance in both speed and error control. The algorithm is now available in the *AIPS++* `qimager` tool (in *AIPS++* versions 1.9.299 onwards).

The image- and uv-facets algorithms remain roughly competitive with w projection only at very low dynamic range. At the sensivities to be acheived with EVLA, w projection will be decisively superior. Hence use of this algorithm (and similar algorithms for other problems) will have a substantial impact on the predicted computing costs for both Phase I and Phase II of the EVLA. We intend to recalculate these costs shortly.

The Fourier plane convolution approach can be used to deal with other wide-field effects, including time- and frequency-smearing, polarization leakage, mosaicing, *etc.*. Algorithms for these will be described in a forthcoming memo.

References

Bracewell, R.N., 1984, *Proc. IAU/URSI conference on "Indirect imaging", Sydney, 1983, ed. Roberts, J. A., Cambridge University Press.* 177-183

Clark, B.G., 1980, *Astron. Astrophys.*, **89**, 377–378

Cornwell, T. J., Perley, R. A., 1992, A&A 261, 353

Frater, R.H. and Docherty, I.S., 1980, *Astron. Astrophys.*, **84**, 75–77.

Perley, R.A., and Clark, B.G., EVLA Memo 63, 2003.

Sault, R., Brouw, W.N., and Staveley-Smith, L., *Astron. Astrophys. Suppl.*, **120**, 375.

Acknowledgements

We thank the *AIPS++* Project team for providing a superb environment in which to do this development. TJC thanks Wim Brouw for his patience in explaining simple geometry via email.

Appendix A: AIPS++ simulation script

We include the *AIPS++* script used for these simulations. This is standalone and should work with any version of *AIPS++* later than (and including) 1.9.299.

```
include 'simulator.g';
include 'imager.g';
include 'qimager.g';

const sim:=function(sim=T, ftmachine='mosaic', algorithms='dirty',
    facets=5)
{

  include 'logger.g';
  dl.purge(0);
```

```
include 'webpublish.g';

cell:="60arcsec"; imsize:=1024;

testdir := spaste('sim-', ftmachine, '-', facets);

if(sim) {
  note('Cleaning up directory ', testdir);
  ok := shell(paste("rm -fr ", testdir));
  if (ok::status) { throw('Cleanup of ', testdir, 'fails!'); };
  ok := shell(paste("mkdir", testdir));
  if (ok::status) { throw("mkdir", testdir, "fails!") };
}

include 'catalog.g';
if(is_fail(dc.list(testdir))) sim:=T;

resultsdir := spaste(testdir, '/results');
webpub:=webpublish(resultsdir,
   'AIPS++ simulation of wide field VLA observations and processing.</p>');
if(is_fail(webpub)) fail;

webpub.comments('This simulates processing of wide-field VLA observations.</p>');
webpub.comments(spaste('Using ', ftmachine, ' FTMachine'));


msname   := spaste(testdir, '/sim.ms');
simmodel := spaste(testdir, '/sim.model');
simsmodel:= spaste(testdir, '/sim.smodel');
simtemp  := spaste(testdir, '/sim.temp');
simpsf   := spaste(testdir, '/sim.psf');
simempty := spaste(testdir, '/sim.empty');
simcl    := spaste(testdir, '/sim.cl');

dir0 := dm.direction('j2000',  '13h0m0.0', '45.00.00.00');
reftime := dm.epoch('iat', '2001/01/01');

if(sim) {

  include 'cscatalog.g';
  cs:=cscatalog();
  cl:=cs.querydirection(dir0, sr='5deg', catalog='WENSS', fluxrange='2Jy');
  if(is_fail(cl)) {
    print "Using old component list";
    cl:=componentlist('sim.cl');
  }
  cl.sort();
  cl.rename(simcl);
  cl.done();

  note('Create the empty measurementset');

  mysim := simulator();

  mysim.setspwindow(row=1, spwname='4BAND', freq='74MHz',
    deltafreq='0.0125MHz',
    freqresolution='0.0125MHz', nchannels=8,
      stokes='RR LL');

    posvla := dm.observatory('vla');
#
#  Define VLA C array by hand, local coordinates
#
    xx := [41.1100006,134.110001,268.309998,439.410004,644.210022,880.309998,
   1147.10999,1442.41003,1765.41003,-36.7900009,-121.690002,-244.789993,
   -401.190002,-588.48999,-804.690002,-1048.48999,-1318.48999,-1613.98999,
   -4.38999987,-11.29,-22.7900009,-37.6899986,-55.3899994,-75.8899994,
   -99.0899963,-124.690002,-152.690002];
    yy := [3.51999998,-39.8300018,-102.480003,-182.149994,-277.589996,-387.839996,
   -512.119995,-649.76001,-800.450012,-2.58999991,-59.9099998,-142.889999,
   -248.410004,-374.690002,-520.599976,-685,-867.099976,-1066.42004,77.1500015,
   156.910004,287.980011,457.429993,660.409973,894.700012,1158.82996,1451.43005,
   1771.48999];
    zz := [0.25,-0.439999998,-1.46000004,-3.77999997,-5.9000001,-7.28999996,
   -8.48999977,-10.5,-9.56000042,0.25,-0.699999988,-1.79999995,-3.28999996,
   -4.78999996,-6.48999977,-9.17000008,-12.5299997,-15.3699999,1.25999999,
   2.42000008,4.23000002,6.65999985,9.5,12.7700005,16.6800003,21.2299995,
   26.3299999];

    diam := array(25.0, 27);
    mysim.setconfig(telescopename='VLA', x=xx, y=yy, z=zz,
    dishdiameter=diam,
    mount='alt-az', antname='VLA',
    coordsystem='local', referencelocation=posvla);
    mysim.setfield(sourcename='M31SIM', sourcedirection=dir0,
   integrations=1, xmospointings=1, ymospointings=1,
   mosspacing=1.0);
    mysim.settimes('10s', '10s', T, '-14400s', '+14400s');
    mysim.create(newms=msname, shadowlimit=0.001,
 elevationlimit='8.0deg', autocorrwt=0.0);

    mysim.done();

    myimager := imager(msname);
    myimager.advise(fieldofview='17deg');
    myimager.setimage(nx=imsize, ny=imsize,
      cellx=cell , celly=cell ,
      stokes="I" , fieldid=1, facets=1);
    myimager.make(simmodel);
    myimager.make(simempty);

    myimager.setoptions(ftmachine="ft");
    myimager.ft(model=simempty, complist=simcl);
    myimager.done();

    t:=table(msname, readonly=F);
    md:=t.getcol('MODEL_DATA');
    t.putcol('DATA', md);
    t.putcol('CORRECTED_DATA', md);
    t.close();
  }

  if(ftmachine=="mosaic") {
    myimager := qimager(msname);
  }
```

```
else {
  myimager := imager(msname);
}
myimager.setimage(nx=imsize, ny=imsize, cellx=cell, celly=cell,
  stokes="I" , fieldid=1, facets=facets);
myimager.setoptions(ftmachine=ftmachine);
myimager.weight('briggs', rmode='norm');

for (algorithm in algorithms) {

  simimage := spaste(testdir, '/sim.', algorithm);
  simrest  := spaste(testdir, '/sim.', algorithm, '.restored');
  simresid := spaste(testdir, '/sim.', algorithm, '.residual');
  simerror := spaste(testdir, '/sim.', algorithm, '.error');

  tabledelete(simrest);
  tabledelete(simresid);
  tabledelete(simimage);
  tabledelete(simerror);

  if(algorithm=='mfclark') {
    myimager.setmfcontrol(cyclefactor=1.5);
    myimager.clean(algorithm='mfclark', niter=10000, gain=0.1,
   displayprogress=F,
   model=simimage, image=simrest, residual=simresid);
  }
  else if(algorithm=='wfclark') {
    myimager.setmfcontrol(cyclefactor=1.5);
    myimager.clean(algorithm='wfclark', niter=10000, gain=0.1,
   displayprogress=F,
   model=simimage, image=simrest, residual=simresid);
  }
  else if(algorithm=='cs') {
    myimager.setmfcontrol(cyclefactor=1.5);
    myimager.clean(algorithm='cs', niter=10000, gain=0.1,
   displayprogress=F, threshold='10mJy',
   model=simimage, image=simrest, residual=simresid);
  }
  else if(algorithm=='mfmultiscale'){
    myimager.setscales(uservector=[0, 6, 12]);
    myimager.clean(algorithm='mfmultiscale', niter=1000, gain=0.7,
   displayprogress=F,
   model=simimage , image=simrest, residual=simresid);

  }
  else if(algorithm=='mfentropy'){
    myimager.setmfcontrol(cyclespeedup=1);
    myimager.mem(algorithm='mfentropy', niter=30, displayprogress=F,
 model=simimage , image=simrest, residual=simresid);
  }
  else if(algorithm=="dirty") {
    myimager.makeimage("corrected", simimage);
  }
  else {
    myimager.make(simempty);
    myimager.residual(model=simempty, image=simimage);
  }
  if(tableexists(simrest)) {
      if(!tableexists(smodel)) {
myimager.makeimage("psf", simpsf);
bmaj:=F; bmin:=F; bpa:=F;
myimager.fitpsf(simpsf, bmaj, bmin, bpa);

myimager.smooth(simmodel, simsmodel, F, bmaj, bmin, bpa, normalize=F);
      }

      imagecalc(simerror,
spaste('"', simrest, '" - "', simsmodel, '"')).done();
    }
    webpub.comments(spaste('<p>The images for ', algorithm, ' image processing</p>'));
    for (name in [simimage, simrest, simresid, simerror]) {
      if(tableexists(name)) webpub.image(name, name);
    }
    webpub.flush();
  }

  webpub.comments('<p>The model, smoothed model, and psf</p>');
  for (name in [simmodel, simsmodel, simpsf]) {
    if(tableexists(name)) webpub.image(name, name);
  }
  webpub.flush();

  myimager.done();

  webpub.comments('<p>Processing script and log</p>');
  webpub.script('wf.g');
  webpub.log();
  webpub.done();
}

#
# Make all the images we need, each in its own directory
#
for (f in [8,16,32,64,128,256]) {
  sim(ftmachine='mosaic', sim=F, algorithms="cs", facets=f);
}
for (f in [3, 5, 7, 9, 11, 13, 15]) {
  sim(ftmachine='wfmemoryft', sim=F, algorithms="wfclark", facets=f);
}
```

NRAO, PO Box 0, Socorro, NM, 87801

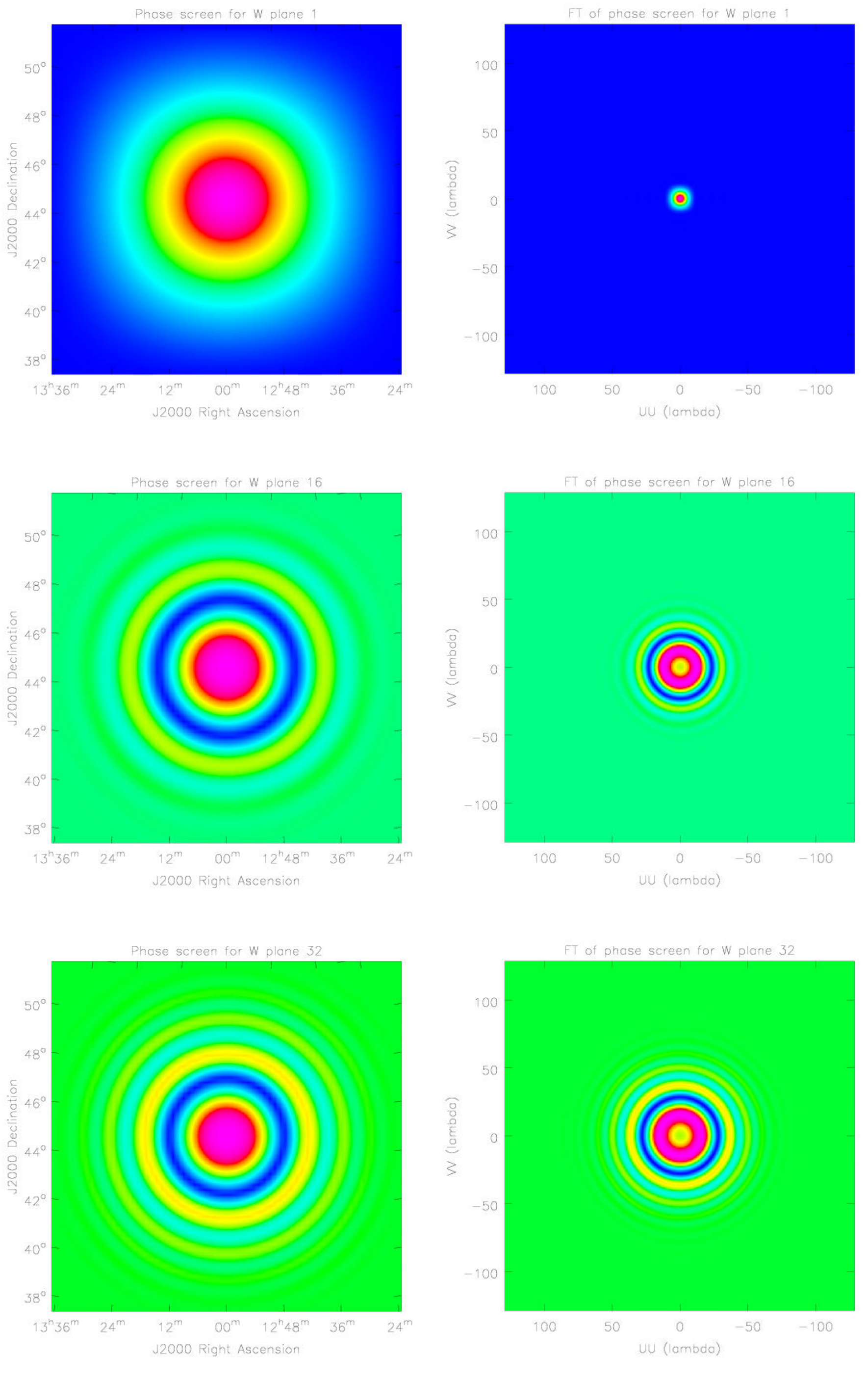
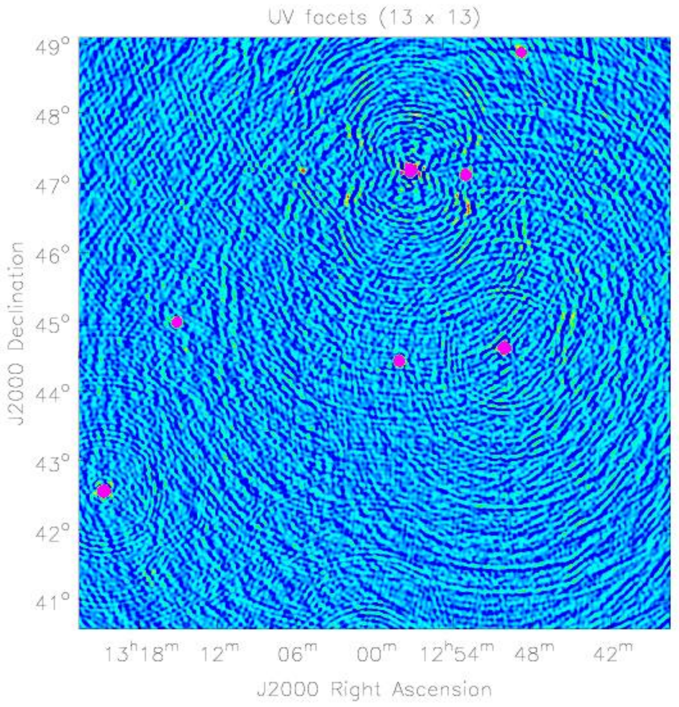


FIGURE 3. Image plane function and Fourier transform for (top) $w = 0$, (middle) $w = w_{max}/2$, (bottom) $w = w_{max}$. The range of brightness is -1 to +1.



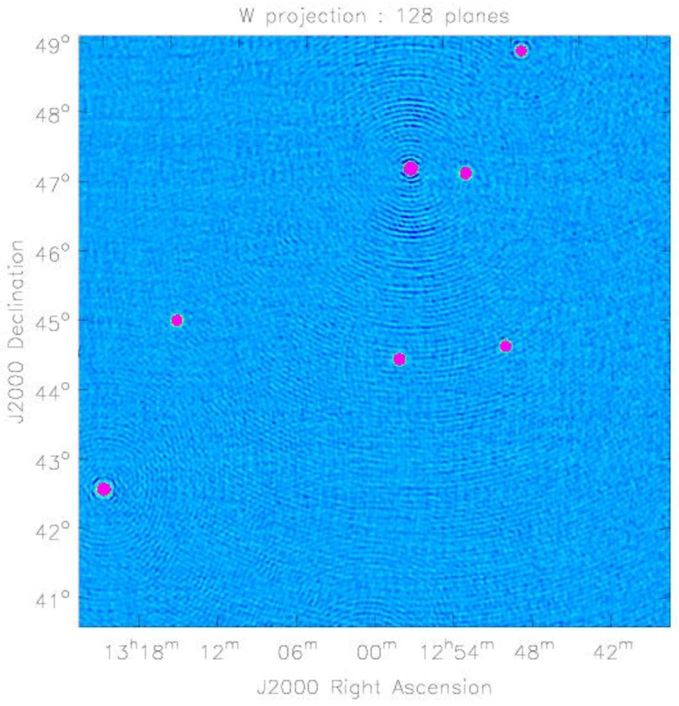


FIGURE 4. Clean images for 74MHz simulation. Top: uv facets (13 x 13), Bottom: w projection (128). The brightness range is -5 to +50 mJy/beam, and the peak brightness should be 16.2Jy. The peak sidelobes seen around the brightest sources in the uv facets image are about 0.3%. Calculation of these images took 2h58m and 28m15s respectively.