Chapter 19

Primary Beams and Mosaicing

19.1 Primary Beams and Primary Beam Correction

In interferometry, the image formed normally by the imaging and deconvolution steps is a representation of the sky *multiplied by the primary beam response* of the antennas. The primary beam is typically similar to a Gaussian function, although it also has sidelobes.

MIRIAD tasks which require knowledge of the primary beam response of a telescope use built-in models of the responses of various telescopes (e.g. ATCA, VLA, Hat Creek, WSRT) – the primary beam model used is determined by the '**telescop**' item or variable. Currently these models assume the primary beam is circularly symmetric and time independent. The task **pbplot** produces some information and can make a plot of the primary beam models.

If you wish to override *MIRIAD*'s model, or if *MIRIAD* does not have a model of the telescope of interest, you may set the primary beam associated with an image or visibility dataset to be a Gaussian of a particular size. This is done by setting the **pbfwhm** item using **puthd**. This gives the FWHM width of the primary beam, in arc seconds, at the reference frequency (the 'reference frequency' for a visibility dataset is the frequency of the first channel imaged!!). For example, to set the primary beam size of the image **lmc.map**, use:

PUTHD	
in=lmc.map/pbfwhm	Set **pbfwhm** of **lmc.map**.
value=1200.0	Set the primary beam FWHM to 20 arcmin (1200 arcsec).

Although there are a few exceptions (e.g. **mfspin** and **ellint**), *MIRIAD*'s analysis tasks do not correct for primary beam attenuation automatically. The task to correct an image for the primary beam is **linmos**. To use **linmos** for this function, you need only set the input and output dataset names. For example

LINMOS	
in=lmc.map	Image to primary beam correct.
out=lmc.pbcorr	Output, corrected, image.
options	Leave unset.

19.2 Mosaicing

The primary beam limits the size of an object that we can observe with a conventional experiment. To circumvent this, a large object can be observed using multiple pointings – this is the practise known as mosaicing. In interferometry, mosaicing is not simply the practise of pasting together multiple tiles of the sky. In interferometry, the adjacent pointings are not independent, and so we can get fundamentally

Table 19.1: Mosaic grid spacing for ATCA dishes

Frequency ν (GHz)	Pointing Spacing $\theta_{\rm hex}$ (arcmin)
1.38	19.6
2.378	11.4
4.8	5.6
8.64	3.1

better images by processing the different pointings together. This is particularly so for extended emission or when the signal-to-noise ratio is low. A description of the theory behind mosaicing can be found in the NRAO Synthesis Imaging Summer School (Lecture 15 – Wide Field Imaging III: Mosaicing – by Tim Cornwell). Other notable references are Cornwell (1988) (Astronomy and Astrophysics, 143, 77) and Cornwell, Holdaway and Uson (1993) (Astronomy and Astrophysics, 271, 697).

19.3 Mosaicing Observing Strategies

The job of planning a mosaic experiment requires extra thought over a simple conventional observation. Issues that you must decide in the planning of an experiment include:

- Pointing grid pattern: In a mosaic experiment, you observe a number of pointings – possibly a few to several hundred, depending on the size of the source of interest. To consider how dense the sampling grid needs to be, consider the primary beam of an antenna. In the $u-v$ plane, the Fourier transform of the primary beam pattern is just the cross-correlation between two antenna illumination patterns. For the 22-m ATCA dishes, the Fourier transform of the primary beam pattern will be of finite and circular extent, having a diameter of 44-m. Because it is of finite extent, Nyquist's sampling theorem indicates that, provided we do not sample in the sky domain coarser than some limit (i.e. provided the pointing grid pattern is sufficiently fine), all information can be retrieved. Assuming a standard, rectangular grid, the sky plane Nyquist sampling limit is

$$\theta_{\rm rect} = \frac{\lambda}{2D}$$

  (λ is the wavelength, and D is the dish diameter). For a well-illuminated dish, this spacing corresponds roughly to half-power point spacing between field centres. Because the extent of the transform is circular, we can do somewhat better than this, by using a so-called hexagonal grid. This grid places pointing centres at the vertices of equilateral triangles – packing six triangles together gives a hexagon. An extension of Nyquist's theorem indicates that

$$\theta_{\rm hex} = \frac{2}{\sqrt{3}}\frac{\lambda}{2D}.$$

  So a hexagonal grid allows a given area of the sky to be covered in a smaller number of pointings (it does also require slightly longer drive times between pointings – see below – which may occassionally be a consideration). Table 19.1 gives this grid spacing for ATCA dishes.

- Dwell time: Most mosaiced experiments will continually switch between the different pointing centres (or a subset of them, if there are too many pointing centres to visit in a single observation). Normally they will be visited in a raster scan fashion. Switching to a new pointing centre typically results in 0.5 to 4 seconds of 'lost' time while the antennas are slewing to the new pointing. This time can be a significant consideration in some experiments – e.g. if the integration time was 10 seconds, and the pointing centre was switched every integration, up to about 40% of the observing time could be lost. To avoid this, you will want to dwell on a given pointing centre for as long as reasonable. This must, however, be traded against loss of tangential $u-v$ coverage that occurs when each pointing is not visited sufficiently frequently. To determine the balance, recall that a correlation does not measure the value of a single point in the $u-v$ plane, but a region corresponding to twice the diameter of the dishes. At transit (when the projected baselines are moving fastest), the time taken for a baseline to rotate to a completely independent visibility point is

$$\tau = \frac{86400}{2\pi}\frac{2D}{L} \text{ seconds}$$

Here L is the maximum baseline length of interest when imaging and D is the dish diameter. Ideally you will want to sample twice as frequently as this, i.e. for N pointings, a dwell time of $\tau/2N$ would be best. You may, however, decide to suffer tangential holes in the $u-v$ coverage.

- Field Naming Convention: When preparing the observe files for an ATCA mosaic experiment, you will create a 'mosaic file'. This gives a field offset, integration time and field name for each pointing centre. To simplify a step in the reduction process (the splitting step only), it is recommended that field names be composed of two parts, separated by an underscore character. The first part should be common to all fields. Typically this will be the name of the object being mosaiced. The second part is unique to each field, typically being a field number. For example, the field name for pointing 123 for a Large Magellanic Cloud mosaic would be called `lmc_123`.

19.4 Visibility Processing

The flagging, splitting and calibration of a mosaic experiment are rather similar to a conventional experiment. The following comments only on any differences. This section will assume that, as is usual, only a single phase calibrator is used for all pointings.

- *Flagging:* Flagging differs only in that, when the instrument is continually changing to a new pointing centre, it can be more difficult to spot the outliers which indicate bad data.
- *XY Phase Correction and Splitting:* XY phase correction is identical with a conventional reduction. Again, as with conventional reduction, for calibration and imaging purposes, it is easiest in *MIRIAD* to split the data into datasets containing a 'single source' and single frequency band. For a mosaic experiment, all the pointings from the object of interest should be considered a 'single source'. As the ATCA on-line system labels each pointing with a different field name, **uvsplit** would normally break the dataset into one file per pointing. To avoid this, use **uvsplit**'s mosaic option. This causes **uvsplit** to use the source name up to any underscore character. Thus, assuming you have used the field naming convention of Section 19.3, **uvsplit** will not split apart all the separate pointings.

  If you failed to follow the naming convention, the **select** keyword of **uvaver** and **uvsplit** can be used to split the dataset up. This is left as a tiresome exercise for the reader.

  For example, assuming a dataset containing calibrators and an observation of the LMC (with appropriately named fields), the following inputs will generate datasets for the calibrators and a single dataset for all the LMC data.

UVSPLIT	
vis=mosaic.uv	The input dataset.
options=mosaic	Assume mosaic experiment, so create multi-pointing output where necessary.

- *Data Selection:* With mosaic visibility datasets, the **select=ra** and **select=dec** selection subcommands can be useful to extract or manipulate a subset of the visibility data.
- *Calibration:* Determining the calibration solutions does not differ from that described in Chapter 11. You will then use **gpcopy** to copy the calibration tables to the multi-pointing dataset.

19.5 Summary of Imaging Strategies

There are two quite distinct methods for reducing a mosaic experiment – the so-called "joint" and "individual" approaches. Although hybrid approaches are also conceivable, they will not be discussed.

The joint approach, which is the simplest, takes advantage of *MIRIAD*'s mosaicing software. In this case, all pointings are handled simultaneously by the imaging and deconvolution software. With the individual approach, a mosaic experiment is treated like a large number of conventional observations, where each pointing is imaged and deconvolved separately. In this case you, the user, are responsible for keeping track of all the pointings. Only as a final step are all the pointings pieced together.

The advantage of the joint approach is speed and simplicity. Also because the deconvolution of all the pointings is done together, it can produce fundamentally better deconvolutions. This is particularly so for low signal-to-noise ratio mosaics and for extended emission (emission comparable in extent to the primary beam – see Cornwell's papers for the argument). It is the approach normally used. However there are disadvantages – the joint approach depends more critically on the model of the primary beam. Errors in the model of the primary beam will tend to be amplified by this approach, particularly when the $u-v$ coverage is poor. Generally the joint approach will be limited to dynamic ranges of several hundred or so.

A more practical difference between the two approaches is that the joint approach generally uses significantly less disk space (this can be an order of magnitude or more for spectral line experiments). However, because the joint approach does all pointings simultaneously, it does use significantly more computer memory in its reduction steps. With current computers, the joint approach is not possible for full resolution ATCA images (6 km array) if you have more that a few pointings.

19.6 The Joint Approach

19.6.1 Theory

For the joint approach, the reduction proceeds in a fashion which appears very similar to conventional observations. First a dirty image is formed (with associated point-spread function), and then a deconvolution algorithm is used to 'clean' this dirty image. Finally the 'restore' step is performed. There are, however, substantial differences – although these are largely hidden from the user.

The task to form the dirty image is still **invert**. The dirty image is formed by imaging (using a conventional algorithm) each of the pointings separately. These individual pointing images are them combined in a linear mosaicing process. This linear mosaicing simply consists of a weighted average of the pixels in the individual pointings, with the weights determined by the primary beam response and the expected noise level. The resultant output dirty image is thus an image of the entire region mosaiced.

The weights are computed to minimise the noise in the resultant image as well as to correct for primary beam attenuation. The output image, $I(\ell,m)$, is given by

$$I(\ell,m) = g(\ell,m)\frac{\sum_i P(\ell-\ell_i,m-m_i)I_i(\ell,m)/\sigma_i^2}{\sum_i P^2(\ell-\ell_i,m-m_i)/\sigma_i^2}.$$

Here the summation, i, is over the set of pointing centres, (ℓ_i,m_i). $I_i(\ell,m)$ is the image formed from the i'th pointing, and $P(\ell,m)$ is the primary beam pattern. The expected noise variance in the i'th field is σ_i^2.

Primary beam attenuation is only corrected for within limits. Because there are large variations in sensitivity across a mosaiced image (the edges of a mosaiced region will have low sensitivity), the imaging software does not always attempt to fully correct for primary beam attenuation. Instead, it constrains the weights so that the noise level does not exceed a certain limit (this limit is based on the noise in individual pointings). This results in some residual primary beam attenuation at the edges of a mosaic or in holes in the pointing grid. This is done by the function $g(\ell,m)$. It normally has a value of 1, but its value drops towards 0 at edges or holes. In this way, the noise level across a mosaiced image is crudely uniform.

Task **invert** also applies geometric corrections to account for the fact that the sky is not a plane. For an east-west array, such as the ATCA, these corrections are exact, meaning that the coordinate geometry of the resultant images is also (nominally) exact. For other array types, the corrections are optimal in the

Table 19.2: Size of Main Lobe of the Primary Beam for ATCA dishes

Frequency ν (GHz)	Primary Beam Main Lobe Size θ (arcmin)
1.38	70.2
2.378	44.5
4.8	20.6
8.64	12.2

sense that they are the best approximation that still results in a convolution relationship (in the sense that such arrays obey a convolution relationship!).

Because the $u-v$ coverage of the different pointings will not be identical, the synthesised beam patterns will differ between pointings. This, and the weighted average process, means that the point-spread function of the resultant dirty image is position-dependent. As most deconvolution algorithms assume a position-independent point-spread function, a conventional algorithm cannot be used. However the point-spread function from the linear mosaicing process is still reasonably compactly described and readily computed. The beam dataset that **invert** produces is not a normal one – it is a cube of beam patterns, one for each pointing. Given this, and some information stored in a auxiliary mosaic information table, the deconvolution tasks can compute the true point-spread function at any position in the dirty image. Being able to compute a point-spread function (or rather, being able to compute a dirty image, given a trial deconvolved image) is the difficult part of writing a deconvolution task. A maximum entropy-based deconvolution algorithm is readily implemented.

The practicalities of this processing are now described.

19.6.2 Imaging – INVERT

Most of the inputs to **invert** are the same as with conventional imaging. Only mosaic-specific considerations will be mentioned here. See Chapter 12 for more information. Note that **invert** supports multi-frequency synthesis, Stokes and spectral imaging.

- **options=mosaic**: The most important thing to remember is to invoke **invert**'s mosaic mode! This causes **invert** to expect multiple pointings in the input visibility data, and to perform the linear mosaicing steps and geometric corrections.
- **options=double**: If you intend to deconvolve, **options=double** should always be used. This is because the full field of the each individual pointing is potentially filled with emission.
- **vis**: When mosaicing, **invert** handles input datasets which contain multiple pointing centres.
- **imsize**: In mosaic mode, this is interpreted as the image size, in pixels, of each subfield. There are two constraints that are important if you wish to deconvolve. These can be relaxed, with corresponding degradation in deconvolution.
  - Ideally **imsize** should be large enough to contain all emission in the main lobe of the primary beam (as *MIRIAD* only models the main lobe, making it larger has no beneficial effect). Table 19.2 gives these sizes as a function of frequency for the ATCA.
  - The image size ***should not*** be a power of 2, or a number within the range (approximately) $[0.9\times 2^n, 2^n]$ (note that just 1 pixel more than a power of 2 is fine – and indeed good). This restriction is to help reduce the effects of the aliasing caused by a "grid-and-FFT" imaging algorithm which **invert** uses.
- **offset**: This has a different meaning in mosaic mode. It gives the position on the sky (the so-called tangent point), in RA and DEC, which is used for geometry calculations. The value is given in the form *hh:mm:ss,dd:mm:ss*, or as decimal hours and degrees. Normally you can allow this to default, and **invert** will choose a central pointing centre as the tangent point.

Typical inputs to **invert** would be:

INVERT	
vis=lmc.uv	The input multi-pointing dataset.
options=mosaic,double	Use mosaic mode and make large beam.
offset	Usually can leave blank.
map=lmc.map	Output image name.
beam=lmc.beam	Output beam name.
cell=	Set cell size.
imsize=	Set output image size.

19.6.3 Deconvolution and Restoration

MIRIAD contains two tasks to deconvolve the mosaiced dirty images produced by **invert**. In terms of theory, practical use and indeed internal implementation, these tasks are quite similar to the deconvolution tasks described in Chapter 13. The major difference is that the 'convolution' operation (which turns a prospective model into a dirty image) is somewhat more involved. Also account must be made of the changing noise level across the dirty image.

The two mosaic deconvolution tasks are **mosmem**, which implements a maximum-entropy-based deconvolution algorithm, and task **mossdi**, which uses a Steer, Dewdney & Ito (SDI) CLEAN algorithm. Generally **mosmem** is superior. Task **mossdi** can be very slow for all but very extended emission, and its results can be poor. Note that, although you can make mosaiced, multi-frequency synthesis images with **invert** (and, indeed, produce a mosaiced, spectral dirty beam), there is no mosaic equivalent to **mfclean**. In deconvolving a mosaiced, multi-frequency image you will have to tactically assume that the spectral index is 0. This should not be a problem – primary beam model errors are probably more significant than spectral errors in these deconvolutions.

If you are deconvolving, note the recommendations for **invert**'s **imsize** parameter, and the use of **options=double**.

If you are familiar with the inputs to the conventional deconvolvers, the inputs to **mosmem** and **mossdi** should be fairly straightforward. In the case of the inputs to **mosmem** and **maxen**, apart from differences in the **options**, the meaning of the **flux** keyword and the default **region**, the only significant difference is in specifying the expected RMS noise level in the dirty image. Because the noise level varies across the dirty image, **mosmem** uses the theoretically expected noise level (which it computes) times a user-specified *fudge factor*, **rmsfac**. That is, if **rmsfac** is set at 1 (the default), then **mosmem** uses the theoretical noise level when calculating its χ^2 statistic.

Typical inputs to **mosmem** are:

MOSMEM	
map=lmc.map	Dirty image produced by **invert**.
beam=lmc.beam	Beam dataset.
model	An initial model estimate – generally unset.
default	The image that the solution should tend towards – generally unset.
out=lmc.model	The output dataset.
niters=30	Maximum number of iterations – default is 30.
region=	Region to deconvolve. The default is the entire image.
measure	Leave unset gives you the Gull measure.
flux=	Estimate of the total flux – its best to give a value.
rmsfac=1	RMS noise fudge factor. Default is 1.
q	An initial estimate of the beams volume. Generally you can leave this unset.
options	Generally leave unset, or
options=doflux	use **doflux** to enforce the flux constraint.

The inputs and use of **mossdi** should be equally simple for someone familiar with **clean**. Given that the task is not recommended, it will not be discussed further.

Having produced a model, we generally want to convolve this with a Gaussian CLEAN beam and add in the deconvolution residuals. This is done by **restor**. The inputs and use of **restor** is identical to a conventional observation (**restor** is the only general task which is smart enough to recognise a mosaiced experiment directly). Task **restor** uses a constant CLEAN beam – it is not a function of position. The only caveat is that, when determining a default CLEAN beam, **restor** fits a Gaussian to the synthesised beam which corresponds to the first pointing. Provided the first pointing is a fairly typical pointing, this will probably be adequate. Otherwise you may wish to use task **mospsf** (see Section 19.6.5 below) to generate an actual point-spread function (at some position) and then use **imfit** to determine Gaussian parameters for it.

Typical inputs to **restor** are:

RESTOR	
map=lmc.map	Dirty image produced by **invert**.
beam=lmc.beam	Beam dataset.
model=lmc.model	Model produced by **mosmem**.
mode	Leave unset to get restored image.
fwhm	Beam size – leave unset to let **restor** fit it, but to the first pointing!
pa	Again leave unset to let **restor** fit it.

19.6.4 Self-Calibration

The software to perform self-calibration is workable and reasonable flexible, although it is rather inelegant. In part, this software has not been upgraded from the time before the 'joint approach' suite was developed in *MIRIAD*.

The self-calibration process is performed in two main steps (there is a minor third step). First the task **demos** ("de-mosaic") is used to break the model produced by **mosmem** into models for individual pointings. That is, **demos** produces many models each one of which corresponds to the nominally true sky multiplied by the primary beam pattern at a pointing. The second step is performed by task **selfcal** (**gpscal** cannot cope with mosaiced observations). Task **selfcal** takes all the models simultaneously and then, for each visibility in the input visibility dataset, it computes a model visibility using the model with the same pointing centre. The observed and model visibilities are then processed by a conventional antenna-gain solver, to produce a table of antenna gains vs time.

In reality, antenna gains will be a function of both time and pointing centre. However **selfcal** assumes that the gains are purely a function of time – not pointing. In practice this should not be a great problem, as time and pointing change together, and integrations that are close in time will also be close on the sky. Note that, short of setting a self-calibration solution interval to be smaller than the integration time, you cannot be sure that a solution interval will contain data from a single pointing.

In the above process, only a subset of all pointings need be used in the self-calibration process. If, for example, you have a strong source in one pointing and all the other pointings have only weak emission, it may well be appropriate to assume that the antenna gains are completely independent of pointing. In this case, the gains can be determined from the one strong field.

We now address the steps in more detail:

1. The **demos** step: This step consists of producing a number of models, one for each pointing. The inputs are described in turn.
   - **map**: This gives the name of the input model image (produced by **mosmem**) to be de-mosaiced.
   - **vis** This will usually be the visibility dataset to be self-calibrated. This dataset is used to determine the pointings present and the primary beam to be used.
   - **select**: This provides normal visibility selection. If only a subset of pointings are being processed, it is convenient to select them here. In this way, models are not generated for pointings that are not of interest. Typically, if you wish to self-calibrate with only a subset of

pointings, you would use the **ra**, **dec** and/or **source** subcommands to select the appropriate ones.

- **out**: This gives a template for the names of the output de-mosaiced models. Task **demos** will generate an output name by appending a number to the template name. For example, the output template `lmc.dmos.` would produce names such as `lmc.dmos.1`, `lmc.dmos.2`, etc.
- **imsize**: This gives the maximum size of the output models. Task **demos** may make the outputs smaller where needed. The default used by **demos** is derived from the primary beam size and the input model, and should be adequate (although if disk-space is tight, you might set a smaller number than that chosen by **demos**).
- **pbtype**: This gives the primary beam type to use in the de-mosaicing process. The default, which is determined from the **vis** dataset, should be adequate.
- **options**: You *must* invoke the option **detaper**. This causes **demos** to account for any residual primary beam attenuation that **mosmem** has left.

Typical inputs to **demos** are:

DEMOS	
map=lmc.model	Model produced by **mosmem**.
vis=lmc.uv	The visibility dataset to be self-calibrated.
select	Leave unset if self-calibrating with all pointings, or
select=source(lmc_123,lmc_124)	select just the fields to be used in the self-calibration process.
out=lmc.dmos.	Output name template.
options=detaper	Account for any residual primary beam attenuation.
pbtype	Generally leave unset.
imsize	Generally leave unset.

2. The **selfcal** step: In general, the inputs to **selfcal** are fairly conventional – see Chapter 14 for more information. There are, however, multiple input models (produced by **demos**) corresponding to each of the pointings to be used in the self-calibration. Note that wildcards will generally make this easy. The other difference *which you must remember* is to use **options=mosaic** to invoke the mosaicing machinery. Note also that **selfcal** will not use a visibility of a particular pointing if there is no model for this pointing. Thus, if you are self-calibrating using only a few of the stronger pointings, you do not have to explicitly select the data for these pointings.

   Typical inputs are:

SELFCAL	
model=lmc.dmos.*	Models produced by **demos**.
vis=lmc.uv	The vis dataset to be self-calibrated.
select	Set as with normal self-calibration.
options=mosaic,phase	Use mosaic mode and phase self-calibration, or
options=mosaic,amp	amplitude/phase self-calibration.

3. Fixing the interpolation tolerance: As noted in Section 11.5, a *MIRIAD* gain table has an associated interpolation tolerance (the time interval over which you can interpolate or extrapolate a gain). Task **selfcal** will set this to the solution interval. If you are self-calibrating with only a few pointing centres, you will want the gains to apply to the entire cycle through the mosaic grid. In this case, you may well want to increase the interpolation tolerance from the default. See Section 11.5 for the details. In summary, you use **puthd** with inputs like:

PUTHD	
in=lmc.uv/interval	Set the 'interval' item of a vis dataset.
value=0.1	Set the tolerance to 2.4 hours (0.1 days).

19.6.5 Some Additional Tools

We briefly describe some other useful tools.

- Listing Mosaic Tables – IMLIST: Task **invert** stores information concerning its linear mosaic operation in the item **mostable** (stored in both the **map** and **beam** datasets). The table can be printed by **imlist**, using **options=mosaic**.
- Mosaic Point-Spread Function – MOSPSF: It is occasionally instructive to look at the point-spread function at a particular position in a mosaic experiment. Task **mospsf** can compute this. Apart from the input beam data-set, the user must specify a position and frequency of interest (the point-spread function is also frequency dependent).
- Mosaic Sensitivity and Gain Images – MOSSEN: Just as the point-spread function varies, so does the expected noise level in an output mosaiced image. Additionally, as mentioned above, **invert** does not attempt to completely correct for the primary beam attenuation where there is too little data (i.e. some primary beam attenuation is remains in the output image). The task **mossen** can produce images of the expected rms noise and the remaining primary beam attenuation given a mosaiced image.

19.7 The Individual Approach

The individual approach images, deconvolves, self-calibrates and restores each pointing separately. It is only when you are happy with the individual images that you would combine them – using a linear mosaicing algorithm.

As noted above, the individual approach is less automated and can produce fundamentally poorer deconvolutions. However in some high-dynamic range applications, it may be preferable. The reason for this is that the deconvolution process does not depend on the primary beam model (nor do some errors such as a constant pointing error affect the deconvolution). With the 'individual approach', it is possible to deconvolve sources (and thus eliminate their sidelobes – which is the important issue) beyond the limit where *MIRIAD*'s primary beam model gives up. To do this, however, you must make images larger than the main primary beam lobe (see Table 19.2 above).

In the following, we assume that the 'joint approach' section has been read and understood.

19.7.1 Splitting and Imaging

Although this is largely a matter of taste, it may be convenient (particularly if self-calibration is to be used) to split the multi-pointing visibility dataset into single pointing ones. Task **uvsplit** (with no options, and only the multi-pointing visibility dataset as input) will do this function. It will also copy across any calibration tables associated with the input dataset.

In the individual approach, you will run **invert** many times, once for each pointing (you may have split the multi-pointing visibility dataset into single pointing ones, or you could use selection by source name to select out the appropriate subset of data). Apart from possibly the names of the input and output datasets, the parameters to **invert** should not be changed between runs.

Even though you are imaging just a single pointing, you will still want to use **invert**'s mosaic mode (**options=mosaic**). This causes **invert** to perform its geometry corrections and to create the the images of the different pointings on the same pixel grid. In this way, no interpolation will be needed when the images from the different pointing are finally combined. Consequently the artifacts and problems associated with interpolation can be avoided.

To compute the geometry, however, *you must provide* a reference position on the sky – the tangent point. The default tangent point is the pointing centre of the data being imaged – *this is not appropriate* as it will vary from pointing to pointing. You will want a tangent point which is the same for all the pointings. Although it can be any arbitrary point, it is best to make it near the centre of the source being imaged.

If there is a point source which dominates the image, you might choose its position as the tangent point to help reduce deconvolution problems. The tangent point is given through the **offset** keyword, in the format *hh:mm:ss,dd:mm:ss* (or as decimal hours and degrees).

As an example, consider an LMC observation, where we wish to image field 123 (which has field name **lmc_123**). Assuming we have a multi-pointing dataset, and wish to use position (RA,DEC)=(4:30,-71:00) as the tangent point. Typical inputs to **invert** would be:

INVERT	
vis=lmc.uv	The input dataset.
select=source(lmc_123)	Select a single pointing.
options=mosaic	Use mosaic mode.
offset=4:30,-71:00	Set position for geometry computation.
map=lmc_123.map	Output image name.
beam=lmc_123.beam	Output beam name.
cell=	Set cell size.
imsize=	Set output image size.

19.7.2 Deconvolution, Restoration and Self-Calibration

Deconvolution is no different to conventional observing. When restoring, you may wish to use the same restoring beam size for all fields. Otherwise treat each pointing as a separate observation.

Much of the discussion in the self-calibration section of the 'joint approach' applies equally well here. The major difference is, of course, that because the deconvolution step has been performed on the individual fields, the **demos** step is not required. If you have split the visibility data into separate single-pointing datasets, you will not need to use **options=mosaic** (although it will not hurt). Also for single-pointing datasets, if you wish to use just one or a few of the stronger fields for self-calibration, you will need to copy the resultant antenna gain tables across to the other datasets, using **gpcopy**.

19.7.3 Image Combination

When you are satisfied with the deconvolution, restoration and self-calibration of all the individual images, task **linmos** can be used to combine them in a linear mosaic. Usually you will just combine the restored images (if you are going to do quantitative analysis on the composite image, it may be best to do a deep CLEAN and use the same restoring beam for all pointings). Although **linmos** can interpolate input images to align them, its algorithm, particularly for geometric correction, is very poor, and so this is *strongly* discouraged. You should use **invert** to make all the input images on the same grid, by setting a common tangent point (**offset** keyword).

Task **linmos** uses the same weighted sum of the input pointings as the 'joint approach' software (see Section 19.6.1). Normally the expected rms noise in the image is determined from the images themselves (image item **rms**). However if this item is missing, or if you wish to override it to get a different weighting, you may enter the expected rms noise via keyword **rms**. Also note that **linmos**, by default, *fully correct for the primary beam attenuation* even when this excessively amplifies the noise. The **taper** option can be used to reduce the correction at the edge of the field, and thus avoid excessive noise amplification.

Task **linmos** can also produce an image giving the expected rms noise as a function of position, and a gain image – see the help on the **options** keyword for these.

Typical inputs to **linmos** are:

LINMOS	
in=lmc_*.cln	Use wildcards to select all images.
out=lmc.mosaic	The output linearly mosaiced image.
rms	Generally left unset.
options	Leave blank to fully correct primary beam,
options=taper	or set to taper at the edge of the mosaic.