Advanced CABB Continuum Data Reduction with MIRIAD

Jamie Stevens (Jamie.Stevens@csiro.au)

September 19, 2012

1 Introduction

This tutorial builds upon the results of the basic CABB continuum data reduction tutorial. The aim of this tutorial is to improve the image quality for the same data. To do this we will use MIRIAD to:

- Image with a spectral dirty beam.
- Perform a multi-frequency clean, and use clean boxes to more closely control the process.
- Use self-calibration to improve the dynamic range.

You should start with the same calibrated data set that is produced by the basic tutorial. The simple imaging described in the basic tutorial is a quick and easy way to check that the data is of reasonably high quality.

2 Reference Image

The image made in the basic tutorial showed a bright central source with a flux of about 27 mJy. After this source was subtracted with imfit, we got a residual image. Let's examine the area immediately surrounding the subtracted source, to estimate the dynamic range of this image.

Using KVIS we can select a box that neatly encompasses an area slightly larger than the central source; I have selected a box with (x, y) SE and NW corners $(606, 122)$ and $(635, 151)$ respectively (these are pixel numbers). We can use the MIRIAD task **imhist** to look at the distribution of flux within this region:

```
miriad% inp imhist
  Task:   imhist
  in       = 2051-377.2868.iresidual/
  region   = boxes(606,122,635,151)
  options  = nbin,100
  cutoff   =
  xrange   =
  device   = /xs
  log      =
```

The output from this task is a histogram plot with 100 bins and a Gaussian fit (shown in Figure 1), and the text:

```
Histogram information
# of points         Mean            rms            Median between
        900    1.0466666E-04  2.1781641E-04     5.0040464E-05 and  7.2479015E-05
Maximum is  1.3941119E-03 at (618,140,1,1) (absolute coordinates)
Minimum is -8.4750168E-04 at (621,137,1,1) (absolute coordinates)
```

The Gaussian fit isn't very good, and that's because the pixels in this area are not random noise. This is probably because the fit was imperfect, and this is probably due to some phase interpolation errors.

Assuming that the rms number quoted by **imhist** is valid, the dynamic range of this image would be $2.69 \times 10^{-2} / 2.18 \times 10^{-4} = 123$. This is not very good.

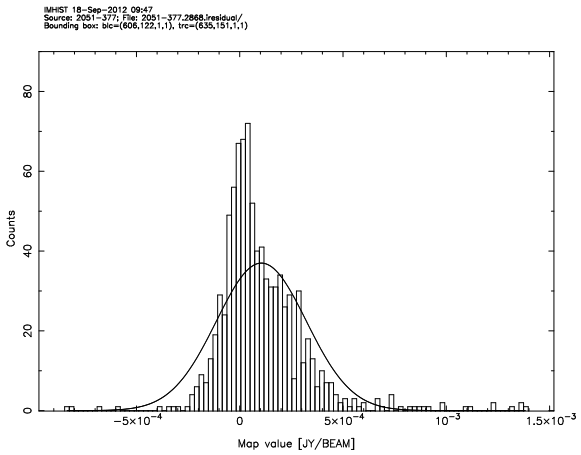

Figure 1: A histogram of the flux distribution in area immediately surrounding the removed central source 2051-377. Also shown is a Gaussian fit made by the task **imhist**.

3 Multi-frequency imaging and deconvolution

The first step we will take to improve our image is to use multi-frequency deconvolution techniques. Normal clean works well when the bandwidth is small, so the size and shape of the dirty beam does not change much across the band, and when the spectral index of the source is close to flat. In this case, neither of those two assumptions can be considered valid.

We need to reimage our field in order to use multi-frequency deconvolution. Not only do we need to create something called the spectral dirty beam (SDB), we need to make our beam three times larger than the field we're attempting to image.

First, we determine the size of the image made in the basic tutorial with **imlist**:

```
miriad% inp imlist
  Task:   imlist
  in       = 2051-377.2868.irestor
  options  = statistics
  region   =
  format   =
  log      =
```

The output of this task tells us that the image is 1234×277 pixels. The pixel size can be determined using **impos**:

```
miriad% inp impos
  Task:   impos
  in       = 2051-377.2868.irestor
  coord    = 1,1
  type     = abspix
  stype    =
  options  =
```

The output of this task tells us that pixel $(1, 1)$ is $(617, 138)$ pixels away from the reference position, and that this offset corresponds to $(521.79, 515.84)$ arcseconds. From this we can see that the pixels are $(521.79/617) \times (515.84/138) = 0.85 \times 3.74$ arcseconds in size.

For normal clean, the beam is required to be twice the size of the region that you want to deconvolve; **invert** has the `double` option to do this automatically, but lacks an option to make the triple-sized beam that the multi-frequency clean task **mfclean** needs. Instead, we will configure **invert** manually to make an image and a beam that are both three times larger than the primary beam:

```
invert% inp
  Task:   invert
  vis      = 2051-377.2868
  map      = 2051-377.2868.mf.imap
  beam     = 2051-377.2868.mf.ibeam
```

```
  imsize   = 3702,831
  cell     = 0.85,3.74
  offset   =
  fwhm     =
  sup      = 0
  robust   =
  line     =
  ref      =
  select   =
  stokes   = i
  options  = mfs,sdb
  mode     =
  slop     =
```

We have kept the `cell` (pixel) size the same as for the basic image, but multiplied the number of pixels in each direction by 3 using the `imsize` parameter. At the same time, we will change from robust weighting to natural weighting (`sup=0`) to minimise the rms noise level in the image, but this will also make the dirty beam worse and perhaps harder to deal with. The only other change is the `options=mfs,sdb`, where we have dropped the `double` option and added the option to create the spectral dirty beam.

We can use **cgdisp** to take a look at the dirty map (Figure 2) and beam (Figure 3):

```
miriad% inp cgdisp
  Task:   cgdisp
  in       = 2051-377.2868.mf.imap
  type     = p
  region   =
  xybin    =
  chan     =
  slev     =
  levs1    =
  levs2    =
  levs3    =
  cols1    =
  cols2    =
  cols3    =
  range    = 0,0,log
  vecfac   =
  boxfac   =
  device   = /xs
  nxy      =
  labtyp   = hms,dms
  beamtyp  =
  options  = wedge,unequal
  3format  =
  lines    =
  break    =
  csize    =
  scale    =
  olay     =
```

There are a number of sources obvious in the dirty map as expected. The beam plot has two panels: this is because there are two planes in the beam. The one on the left (the first plane) is the beam's spatial structure, while the second plane is the behaviour spectrally; **cgdisp** won't display this properly.

We can deconvolve the image now using **mfclean**:

```
miriad% inp mfclean
  Task:   mfclean
  map      = 2051-377.2868.mf.imap
  beam     = 2051-377.2868.mf.ibeam
  model    =
  out      = 2051-377.2868.mf.imodel
  gain     =
```

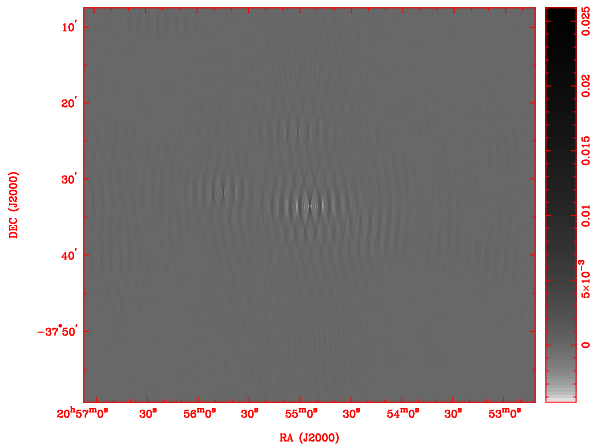

Figure 2: The dirty map made for the 2051-377 field, centred at 2868 MHz. This image is 3 times larger than the primary beam region.

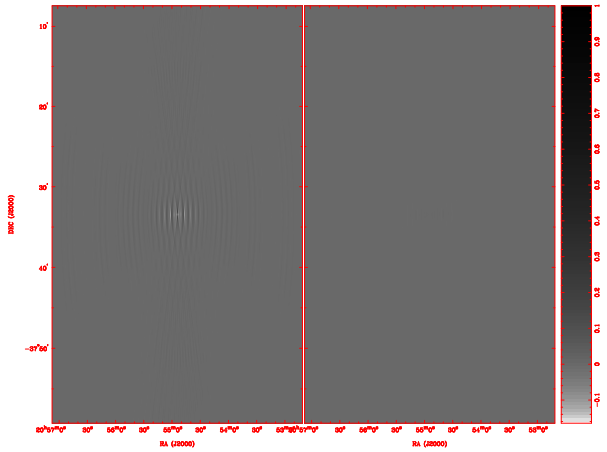

Figure 3: The dirty beam made for the 2051-377 field, centred at 2868 MHz. This beam is 3 times larger than the primary beam region.

```
  cutoff    = 4e-5
  niters    = 1000
  region    = relcenter,boxes(-617,-138,617,138)
  minpatch =
  speed     =
  mode      =
  log       =
```

We choose to clean the entire primary beam region using the `region` parameter. The output of this task is:

```
Starting to iterate ...
 Clark  Iterations: 9
 Residual min,max,rms:   -5.988E-03   1.071E-02   1.358E-03
 Clark  Iterations: 88
 Residual min,max,rms:   -2.372E-03   2.618E-03   5.545E-04
 Clark  Iterations: 1000
 Residual min,max,rms:   -2.455E-03   2.190E-03   3.557E-04
 Stopping -- Maximum iterations performed
```

And after using **restor** to make the clean image, we get the result in Figure 4, and Figure 5 shows the primary beam area only.

Now we can try to measure the source and produce the residual image, to see how much this has improved our dynamic range. We get the following from **imfit**:

```
Source  1, Object type: point
```

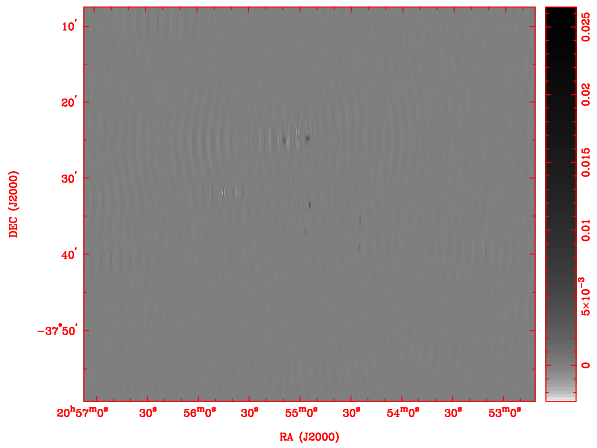

Figure 4: The clean map made for the 2051-377 field, centred at 2868 MHz. This image is 3 times larger than the primary beam region.

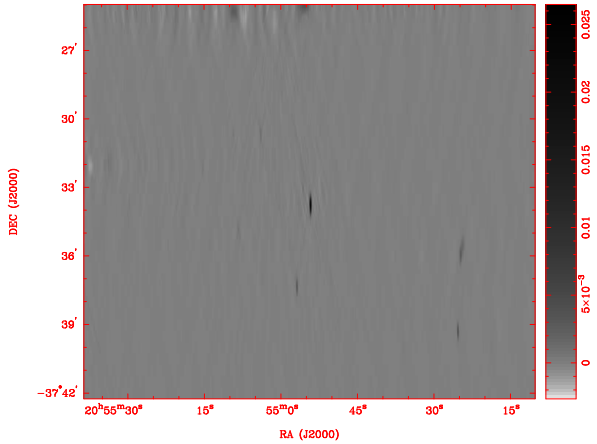

Figure 5: The clean map made for the 2051-377 field, centred at 2868 MHz. This image shows the primary beam region only.

```
  Peak value:                 2.6746E-02 +/-  2.1186E-04
  Offset Position (arcsec):     -2.444    -9.327
  Positional errors (arcsec):    0.015     0.144
  Right Ascension:               20:54:54.194
  Declination:                   -37:33:48.327
```

This is a slightly different flux density than from the basic tutorial. Using **imhist** we can get the residual flux distribution, but we need to recompute the bounding box now that the image is 3 times the size; we can better frame it against the reference pixel:

```
miriad% inp imhist
  Task:   imhist
  in       = 2051-377.2868.mf.iresidual/
  region   = relcenter,boxes(-11,-16,18,13)
  options  = nbin,100
  cutoff   =
  xrange   =
  device   = /xs
  log      =
```

The plot of the residual flux is shown in Figure 6, and the output is as follows:

```
Histogram information
# of points         Mean            rms            Median between
        900   -6.7412366E-06  2.3315370E-04    -3.0824151E-05 and -7.1248505E-06
```

```
Maximum is  1.2939689E-03 at (1852,416,1,1) (absolute coordinates)
Minimum is -1.0735933E-03 at (1855,416,1,1) (absolute coordinates)
```

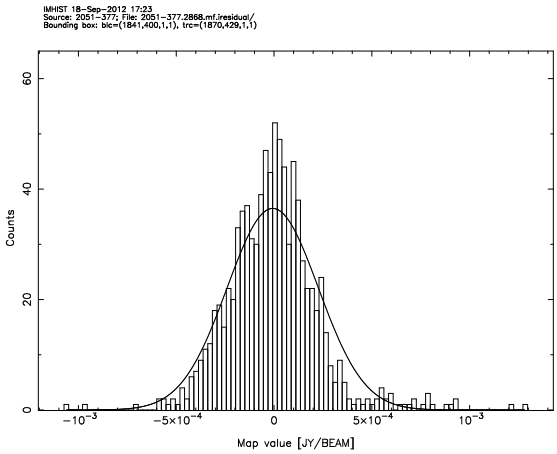

Figure 6: A histogram of the flux distribution in area immediately surrounding the removed central source 2051-377, after multi-frequency cleaning. Also shown is a Gaussian fit made by the task **imhist**.

You might notice that in fact our dynamic range has degraded to 114. What is encouraging though is that the noise appears more Gaussian than before, and the mean has dropped much closer to zero. It appears that cleaning more may improve this situation.

4 Controlling deconvolution

But instead of blindly letting **mfclean** find and deconvolve the sources that are giving us trouble, we can give it some apriori information about the source locations via "clean boxes". We can begin by marking out the brightest sources in the full field now using the **cgcurs** task:

```
miriad% inp cgcurs
  Task:   cgcurs
  in       = 2051-377.2868.mf.irestor/
  type     = p
  region   =
  xybin    =
  chan     =
  slev     =
  levs     =
  range    = 0,0,log
  device   = /xs
  nxy      =
  labtyp   = hms,dms
  options  = region
  3format  =
  csize    =
```

Once you have started this task, the image will appear in the PGPLOT window and some information will appear in the terminal instructing you on how to mark out your regions. You should begin by putting four points around the central source as a rectangle, very close to the source. For each point, simply click the left mouse-button. Once you have surrounded the source, press the `X` key and a line should appear to connect each of the points, completing the region. To begin selecting another region, press `A` and use the mouse again as before. Your selection should include the bright source to the North of the main region, as it sidelobes can be seen in the residual image, so it requires cleaning. When you have finished you may have something that looks like Figure 7. To exit this task press `X` while not selecting a region.

The **cgcurs** task will make a file called `cgcurs.region` in your directory that will contain the region specification. It will use the `poly` specification for the `region` keyword. You may choose to craft your own region using something like KVIS; zoom in a region that neatly encompasses the sources, and determine the pixels at the opposite corners.

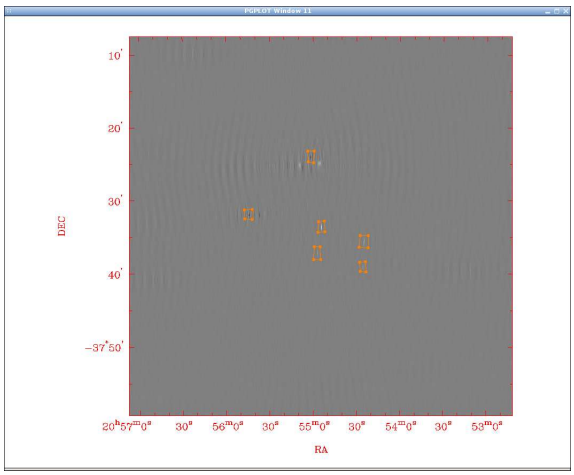

Figure 7: An example of selecting several regions for cleaning with **cgcurs**.

After you've made your region, we can go back to **mfclean** and clean again:

```
miriad% tget mfclean
  Task:   mfclean
  map      = 2051-377.2868.mf.imap
  beam     = 2051-377.2868.mf.ibeam
  model    =
  out      = 2051-377.2868.mf.imodel
  gain     =
  cutoff   = 4e-5
  niters   = 10000
  region   = @cgcurs.region
  minpatch =
  speed    =
  mode     =
  log      =
```

One difference here is that the `region` keyword refers to the file that we just created in **cgcurs**. The other is that now we are using clean boxes, you may choose to do a much deeper clean, to allow the task to go until the rms cutoff is reached.

You may find that when you go to execute this task, it will complain that the region you have selected is too large, and the task will not work. If this happens, you have two choices: select a smaller region to clean, or make a larger image in **invert** and try again.

Once you have run this task, and you are happy that **mfclean** continued to do a good job (ie. the min, max and rms statistics go down at each cycle), take a look at the central region where all the boxes were made. You should see even more sources that may not have been apparent before. You may even choose to go back and put clean boxes around them, as some of them will show sidelobe structure.

One particular region that works reasonably well (on an image/beam that is 4× the primary beam size) is as follows:

```
arcsec,poly(13.26,28.44,-17.46,28.44,-17.46,-46.16,13.26,-46.16)
arcsec,poly(1107.31,-1381.27,105.18,600.11,70.15,604.49,70.17,534.02,96.44,542.85)
arcsec,poly(22.03,37.22,-26.23,41.61,-30.63,-46.16,26.43,-50.54)
arcsec,poly(109.55,617.74,57.01,617.7,61.41,520.81,105.21,529.65)
arcsec,poly(-320.35,-76.35,-386.19,-76.12,-386.34,-176.97,-311.68,-172.85)
arcsec,poly(-316.22,-295.53,-364.54,-290.98,-369.04,-374.17,-320.7,-369.96)
arcsec,poly(57.18,-168.94,8.88,-168.95,4.49,-274.13,61.59,-274.11)
arcsec,poly(631.7,135.82,565.9,139.81,566.07,56.38,627.49,61.14)
arcsec,poly(9.58,29.65,-14.34,29.65,-16.74,-51.68,9.58,-46.89)
arcsec,poly(93.1,609.78,74,604.96,69.25,525.73,95.51,528.16)
arcsec,poly(621.66,141.69,585.79,141.48,588.39,48.14,624.26,50.75)
arcsec,poly(-332.6,-75.03,-373.28,-74.89,-368.63,-175.3,-325.54,-173.06)
arcsec,poly(-330.48,-299.64,-354.44,-304.34,-354.54,-380.72,-332.98,-380.8)
```

```
arcsec,poly(40.71,-183.14,19.16,-185.54,19.17,-254.82,45.51,-257.2)
arcsec,poly(177.07,-51.52,155.53,-51.56,155.56,-94.6,177.1,-94.56)
arcsec,poly(124.31,199.7,105.19,197.28,107.6,154.18,126.73,154.2)
arcsec,poly(186.46,199.8,169.73,197.37,164.98,151.86,184.11,154.29)
arcsec,poly(83.6,429.77,64.5,424.96,62.12,379.39,76.45,384.19)
arcsec,poly(28.71,67.94,7.19,67.94,4.79,20.08,31.11,10.52)
```

This region is available as `advanced_cgcurs.region`, and should come with this tutorial.

With a decent image, redo the measurement of both the flux density of the central source and the corresponding dynamic range. You should be able to get a dynamic range of around 200, and possibly better, but this is still less than ideal.

Let's look more closely at the central region in Figure 8. There still appears to be some emission near where the central point source has been subtracted. This could be another source, but it is more likely that it is due to phase errors that have caused some flux to be misplaced in the image. To correct this and further improve the image, we will need to do self calibration.

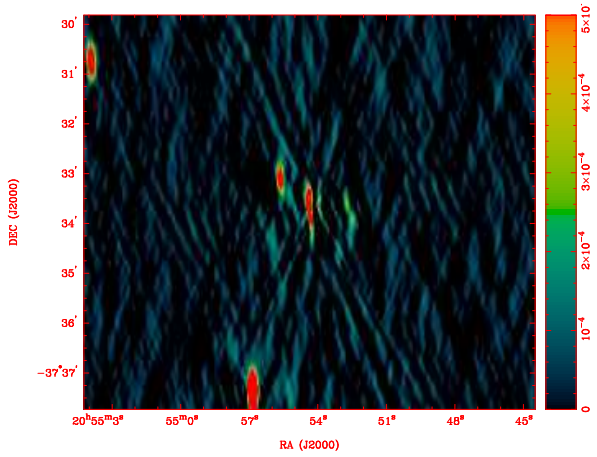

Figure 8: The area surrounding 2051-377 after a point-source fit has been subtracted.

5 Self calibration

The image we have made now has relied on tracking the phase of the instrument via observations of a nearby phase calibrator, and interpolating in time to obtain the phase on this field. This procedure is susceptible to errors.

Phase calibration attempts to determine the instrumental phase and how it changes with time and position. It does this by comparing the observed phases of one or more sources with an ideal model of what the phases should have been. Any differences can be used to determine the instrumental phase.

Now that we have done some initial cleaning, we have a model of the field, as a combination of point source components. This model can now be used to determine the instrumental phases, and because we are using the field we are interested in to calibrate the instrument, we call this process "self calibration".

Self calibration is an iterative procedure, and the first step is to take one or two of the brighter sources in this field and calibrating with them. We have a bright source in the field centre, so we can look at the clean model of roughly the same region as in Figure 8, in Figure 9.

In Figure 9, the central source component is clearly visible, although it is surrounded by some "fluff" that may be the result of phase errors. By selecting only the strongest component and calibrating the phases from it, we may be able to get a better image. We do this using **selfcal**:

```
miriad% inp selfcal
  Task:   selfcal
  vis      = 2051-377.2868
  select   =
  model    = 2051-377.2868.mf.imodel
  clip     = 0.013
  interval = 1
  options  = phase,mfs
```

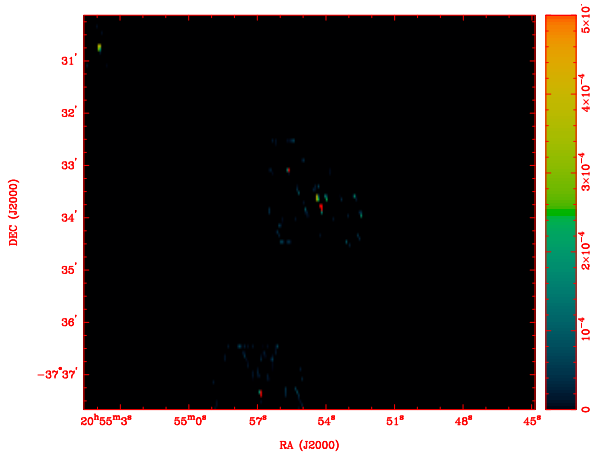

Figure 9: The clean model in the area surrounding 2051-377.

```
minants  =
refant   =
flux     =
offset   =
line     =
```

The `clip` parameter here is set to select only the strongest pixel in the model, and not the surrounding pixels. We are preferentially selecting the position of the source with this method, and it is possible that we are not exactly right. If a couple of adjacent model pixels have similar amplitudes, it could be that the actual source position lies between the pixels, and that our gridding is not fine enough. In this case, you could go back to **invert** and remake the image with a smaller `cell` size. By default, **invert** will try to sample the synthesised beam with three pixels in each direction, but five may give you a better result. For now, in this tutorial, we will continue with the default gridding.

After you have performed the self calibration, you will need to remake the images in **invert** again, and then deconvolve and restore as normal.

6 The rest is up to you!

You now have experience with using the tools that MIRIAD has to offer when imaging. It's up to you to continue with this (or your own) data to get the best image quality that you can.

If you choose to continue with this dataset for this tutorial, here are some tips.

- After a few rounds of self calibration, you might start to realise that in fact the central point source may not be the only thing at the centre of this image...
- With that in mind, using the histogram of the central region may not be the best way to obtain the dynamic range of this image. You could use the `mode=residual` parameter in **restor** to get an image with all the clean components subtracted. If **mfclean** has done a good job, this should be a very empty image with just Gaussian noise. You will probably see other sources in this residual image though, the ones that you didn't put clean boxes around; imaging is most definitely an iterative procedure.
- There's nothing to say that you have to stop cleaning at the theoretical rms noise level. Try cleaning even deeper to see what you get.
- We've been working at the highest frequencies in this band, and there are three other bands to work with and image. You can self-calibrate everything though based on this model, by using the whole band as the input to **selfcal**.
- Imaging the whole 1-3 GHz band at once is probably worth trying at least once, with a well calibrated dataset, just to see what problems you'll face.