THE HIDDEN DEMON IN THE SKY
Mapping high velocity hydrogen clouds above the galactic plane.

Niels Joubert
May 9, 2007[1]
http://ugastro.berkeley.edu/~njoubert/

ABSTRACT

We attempted to use natural hydrogen emission to map a high velocity cloud falling into our galaxy at speeds in the region of 120 km/s to 170 km/s. We went to extensive trouble to track and predict our extended object across the sky, and spent approximately 62 hours on observations alone. Our object had peak signals of about 2 Kelvin, forcing us to use several procedures to minimize noise and tease out our signal from the background radiation and equipment noise. We used Gaussian fitting and Fourier filtering along with the spectrum calibration process to successfully reduce all our observations to useful velocity and intensity values, which we put through an interpolation and resizing process to successfully generate color maps of our object.

[1]Document created Thursday, April 12, 2007

Contents

1. INTRODUCTION

The final laboratory project of the undergraduate radio astronomy course was our first mapping project. We used our previously gained knowledge of the Hydrogen line to observe an extended source, process the observed spectra and generate an image of the hydrogen distribution in an object of interest in the sky.

2. HIGH VELOCITY HYDROGEN CLOUD COMPLEX C

The High Velocity Cloud Complex C can be found in the northern skies circling around the north celestial pole, up above our galactic plane. This huge extra-galactic cloud of hydrogen is being gravitationally accelerated towards us at hundreds of kilometers per second. Several theories exist about the origin of these clouds of gas. Some claim that it is the product of massive supernovae, throwing clouds of hydrogen far out of our galactic plane. An equally provocative theory states that it is hydrogen left over from the original formation of our galaxy. In either case, we find huge clouds of this extragalactic hydrogen above and below our galactic plane. We set out to map a small region of the cloud complex C, in an area where it is known to peak in intensity.

3. DATA ACQUISITION

The process to predict, track and observe our object is much more involved that in any previous project we attempted. In this study, we need to observe a distributed object taking up a fairly big area in the sky. Multiple observations covering the object has several challenges a single observation lacks. With the time of a single observation lasting as long as 45 minutes we cannot observe the whole object in one observing run, so we need to be able to make multiple runs, each run observing the parts of the object we haven't recorded. We also need to track the object across the sky for the duration of each observation. This process involves multiple coordinate transforms, extensive data collection and tracking software and a deft hand at UNIX hacks to ease the process of file manipulation.

3.1. PREDICTION

3.1.1. Coordinate Transformations

We describe our object in Galactic Coordinates - a coordinate system using the galactic plane and poles as reference points. This is an excellent system for describing objects in and around our galaxy, but it takes several transforms to find a point in the Horizontal Coordinate system that we can point to with our telescope. These transformations depend on time and terrestrial location, and is easiest to accomplish using several matrix rotations.

We start by converting Galactic Coordinates (l, b) to Equatorial Coordinates (α, δ). This is the easier transformation, since the rotation matrix between these two coordinate systems is constant - it does not depend on position or time, since the sky is fixed against both coordinate systems involved. We follow the following set of matrix math equations, first converting from spherical to rectangular coordinates, then rotating from galactic to equatorial coordinates, and finally converting back into spherical coordinates. The IDL code is given in the Appendix, section 7.1.

$$\mathbf{X} = \begin{bmatrix} \cos(b)\cos(l) \\ \cos(b)\sin(l) \\ \sin(b) \end{bmatrix} \tag{1}$$

$$\mathbf{R} = \begin{bmatrix} -0.054876 & -0.873437 & -0.483835 \\ 0.494109 & -0.444830 & 0.746982 \\ -0.867666 & -0.198076 & 0.455984 \end{bmatrix} \tag{2}$$

$$\mathbf{X}' = \mathbf{R}\mathbf{X} \tag{3}$$

$$\alpha = \tan\left(\frac{\mathbf{X}_2^{'}}{\mathbf{X}_1^{'}}\right) \tag{4}$$

$$\delta = \sin\left(\mathbf{X}_3^{'}\right) \tag{5}$$

This gives us the right ascension and declination values of a point in the sky. We now want to convert this to an azimuth and altitude for our telescope. This transform is more involved, since the position of the telescope and the time of day both affect this transformation.

We implement the following algorithm to convert from Equatorial to Horizontal coordinates, where ϕ is the station's terrestrial latitude, and LST is the local sidereal time:

$$ha = LST - \alpha \tag{6}$$

$$\mathcal{X} = \begin{bmatrix} \cos\delta\cos ha \\ \cos\delta\sin ha \\ \sin\delta \end{bmatrix} \tag{7}$$

$$\mathcal{R} = \begin{bmatrix} -\sin\phi & 0 & \cos\phi \\ 0 & -1 & 0 \\ \cos\phi & 0 & \sin\phi \end{bmatrix} \tag{8}$$

$$\mathcal{X}' = \mathcal{R}\mathcal{X} \tag{9}$$

$$azimuth = \arctan\left(\frac{\mathcal{X}'[1]}{\mathcal{X}'[0]}\right) \tag{10}$$

$$altitude = \arcsin\left(\mathcal{X}'[2]\right) \tag{11}$$

Applying these procedures to our coordinate box gives us the location of our object as show in Figure 1.

3.1.2. Pointwise Position over Time prediction

The above process takes us from one point in Galactic Coordinates to the same point described in Horizontal Coordinates. This can be described as a mathematical function with galactic latitude and longitude, terrestrial latitude and sidereal time as arguments:

$$(az, alt) = f(l, b, \phi, LST) \tag{12}$$

To simulate the movement of a point across the sky, we fix all the arguments to f except the local sidereal time, which we vary over a predefined range. We can use this to create plots of an object's movement across the sky over the course of a 24 hour period. 24 hours is well suited, since this encompasses a full cycle back to the same point from which we started looking at the object. Why? Because most objects are fixed against the sky, and it is the earth's rotation over 24 hours which causes their apparent motion. This plot shows in an easy glance when our object will be above the horizon, and we can superimpose our telescope's tracking limits onto the plot to get an estimate of object visibility during the day. Figure 2 shows the movement, over 24 hours, of point $(120°, 0°)$.

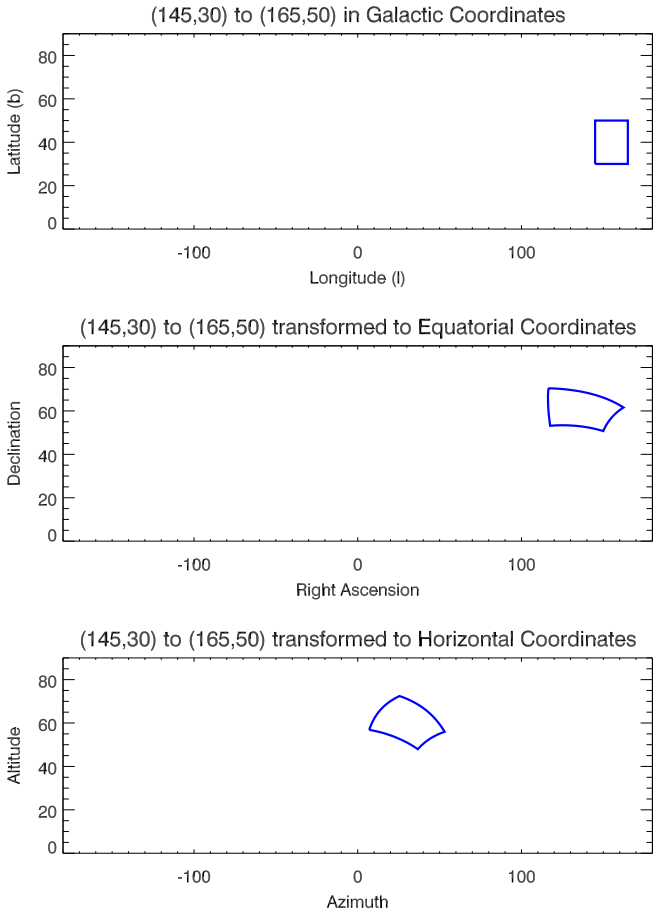

Fig. 1.— The position of the Galactic Coordinate box $(145°, 30°)$ to $(165°, 50°)$ transformed to Equatorial and Horizontal coordinates (as of 18:00 PST May 8th 2007, Berkeley, CA)

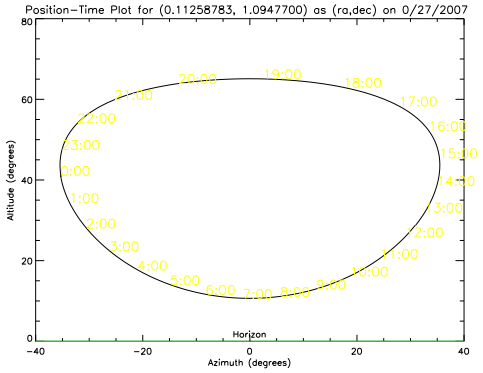

Fig. 2.— The movement of $(120°, 0)$ over 24 hours, plotted against azimuth and elevation.

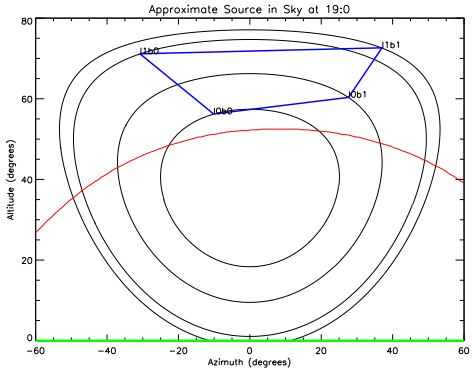

Fig. 3.— A prediction of the paths of the 4 corners of our coordinate box bounded by $(145°, 30°)$ and $(165°, 50°)$, and the approximate source position at 19:00, May 7th 2007, plotted against azimuth and altitude.

3.1.3. Approximate Block Position over Time prediction

Our object is more complex than one point, and we need to simulate the movement of our extended object over time. We can approximate our object to a first degree by modeling it as the region between the 4 galactic coordinates describing the boundary of the square region our object is in. Using the same method as above for all 4 points, connecting each with its two neighbors, allows us to plot the position of our object at any point in time. An example of this program's output is given in Figure 3 To further understand our object's movement across the sky, we iteratively plotted the position of our approximatation, and built an animation of our object's movement across the sky. By overlaying the plots with our radio telescope's pointing limits, we found the approximate times that observation was possible.

We used this method to test the feasibility of our first object of interest - the region around the north celestial pole, where interesting hydrogen shells can be found. Unfortunately our model showed us that our object is almost completely below the telescope's pointing limits at all times, with only one corner rising around 17:00 hours, making this project impossible. Figure 4 shows the position of our original object at various times during the day. The midsection can be seen to never rise above our telescope's limits. Our model saved us several days, since we were able to investigate other possible objects to study before we started using precious telescope time, only to find the telescope limits preventing us from our project. We abandoned that project for the current one of mapping the high velocity cloud complex C within the first week.

3.1.4. Multi-point over Time prediction

Approximations are great for initial investigation, since they are normally faster and easier to implement than an exact model. Now that we have an approximation of our source's movement, we would like to model the object itself. Since our telescope has a finite, 4 degree beam width, we saw no need to model the object as a continuous distribution of matter, but rather as a group of coordinates that we point the telescope to. Our object, covering galactic coordinates $(145°, 30°)$ to $(165°, 50°)$, becomes approximately 100 azimuth/altitude points. The process of creating these points is more involved than a simple division into a square grid, since the rotation between coordinate systems causes a warping of the square into a partial segment of an ellipsoid. The math involved is shown in section 3.2.1 on Tracking. What concerns us in this section is the prediction of the position of these points over time. An example of our program's output can be seen in figure 5. This plot is, incidentally, less useful than the previous approximation, since the amount of points is such that it becomes overwhelming. On the other hand, this can be used effectively to see an accurate distribution

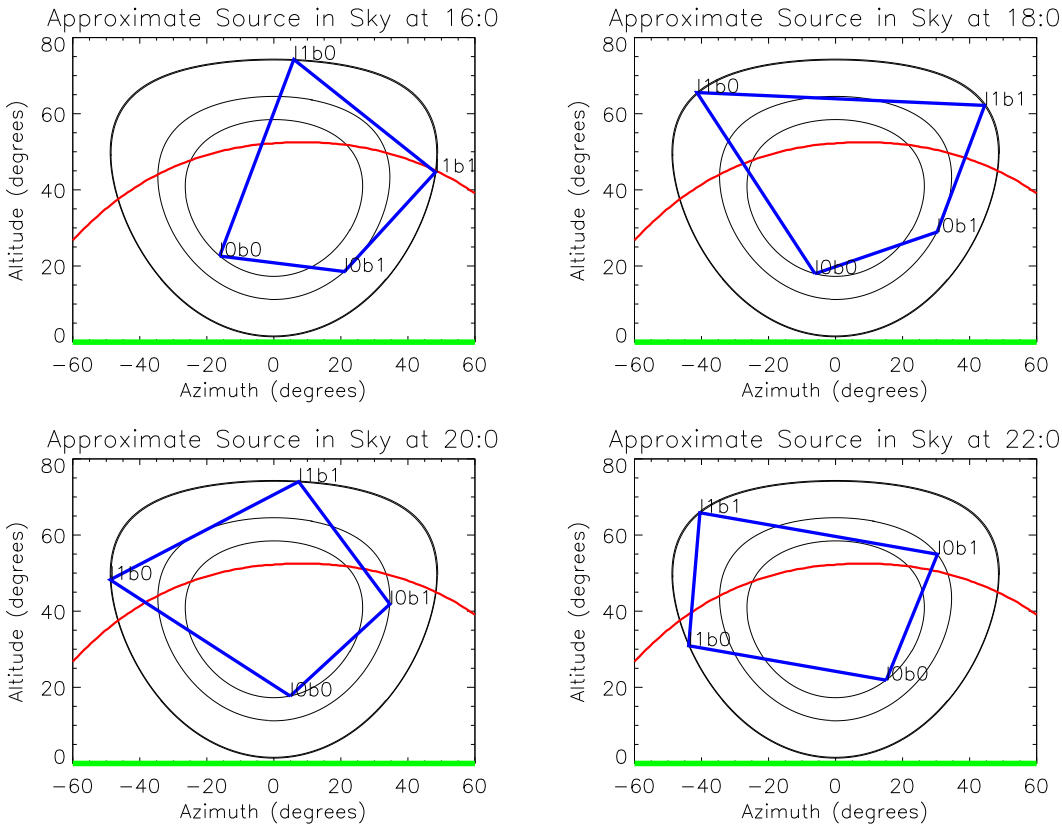

Fig. 4.— The movement of our original object for the time 16:00 to 22:00, showing that it never rises fully above the horizon, making this project impossible with the constraints of the telescope.

of points across our object at one instant in time. We expand on this idea in section 3.2.2 to simulate our actual tracking procedure. We also present a plot in that section of our object's distribution at one point in time.

3.2. TRACKING

Now that we know how our object moves across the sky and that it is visible to our telescope, we want to find the most effective tracking algorithm to observe our object.

3.2.1. Great Circle Distance and Observing a continuous object with discreet observations

Since our object is a continuous spread of hydrogen gas spread occupying a square in Galactic Coordinates, and we have a finite beam width which we point in Horizontal Coordinates, we break our object into discreet parts through the observation process. We can think of the observation process as sampling the object, with each observation contributing one sample spectrum. We want to take observations at the right positions, so that we don't oversample and have duplicate information, or undersample and miss important parts of our object. The reason this process is necessary to discuss, is the geometry of our object when transformed from Galactic to Horizontal coordinates. If we define the box we want to sample over as (l_0, b_0) to (l_1, b_1) , we can define our grid as $(\Delta l, \Delta b)$ with each equal to 2° (since our beam width is approximately 4°, and we sample twice per beam width). This grid would not take into account the warping that occurs in plotting

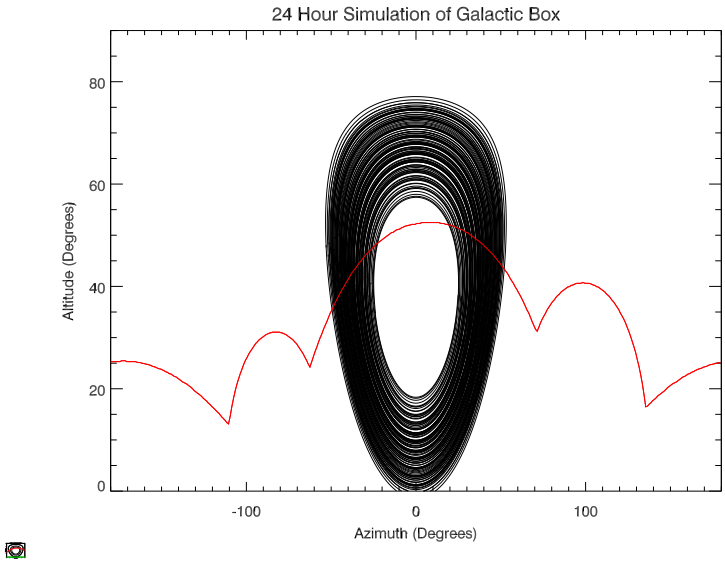

Fig. 5.— The movement of each integer point in the coordinate box $(145°, 30°)$ to $(165°, 50°)$ over 24 hours, plotted against azimuth and elevation.

this rectangle Galactic Coordinate box in Galactic Coordinates, as can be seen in Figure 1. The angular distance between two points along Galactic Latitude does change linearly, but the angular distance between two points along Galactic Longitude depends on the Galactic Latitude of the positions. Thus, we define our coordinate box and grid as follows:

$$\begin{aligned} (l_0, b_0) &\rightarrow (l_1, b_1) \\ \Delta x &= 2° \\ \Delta b &= \Delta c \\ \Delta l &= \frac{\Delta x}{\cos b} \end{aligned}$$

We apply this discretization[2] to our object, and we get a distribution of points equally spaces in Horizontal Coordinate Space, allowing us to observe with as little redundancy as possible.

3.2.2. Telescope Tracking Algorithm simulation

Tracking and observing an extended object becomes an interesting exercise, since each consecutive point is observed slightly later in time than the previous point. This means that we need to model the position of each point in a similar fashion as the previous section on Multi-point prediction, except each point is observed slightly later than the previous one. In our original endeavor, we expected an observation time of approximately 3 minutes per observation. We built a tracking simulation that runs 8 different tracking algorithms across the Galactic Coordinate box to give us an idea of how the discreet data points of our object can be tracked. The output of our program can be seen in Figure 6. We found that, for observations

[2]Discretization concerns the process of transferring continuous models and equations into discrete counterparts. This process is usually carried out as a first step toward making them suitable for numerical evaluation and implementation on digital computers -Wikipedia

that last a small amount of time per point, we can observe our object in one extended scan. Unfortunately, our expected signal peaks at about 3 Kelvin, so we need to observe for as long as 45 minutes per point. This means that such a brute force approach as scanning across the whole object at once will not yield any good results.

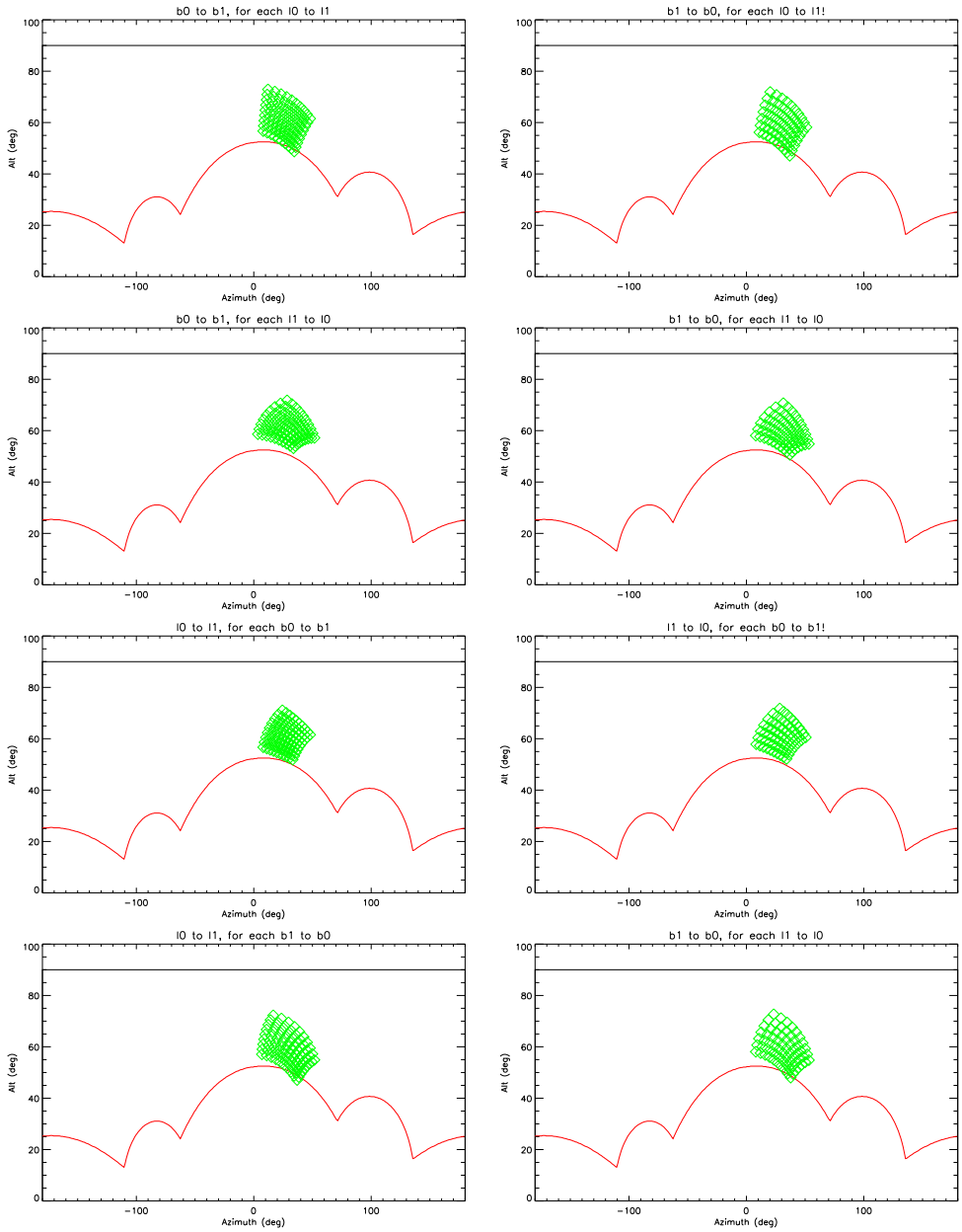

Fig. 6.— The movement of the discreet point in the coordinate box $(145°, 30°)$ to $(165°, 50°)$ over a short period of time. This is modeled by incrementing the time by 2 minutes between plotting of two consecutive points, with each plot moving across the source in a different way.

3.2.3. Multi-observation Tracking

We need to build a level of sophistication into our tracking program to provide for the rising and setting of our object during the extended observation time we need. In section 3.3 we explain the parameters of our observation. At this point, it is enough to mention that each point will be observed for an extended period of time (30 minutes or longer). We need to track the source as it moves during an observation, and we have to take into account that the point we are observing can set in the middle of an observation, and try to minimize the occurance of this event. The fact that we have a minimal amount of telescope time also forced us to think how we can get as much data as quickly as possible. We devised the following algorithm to track our object:

1 Select the next untaken (or partially taken) point with the highest possible altitude of all the untaken points.

2 Start the observation by pointing the telescope.

3 Take the two calibration spectra.

4 Repoint the telescope. If the point has set, discard and go back to 1

5 Loop the following 4 times:

- Take a quarter of the required spectra.
- Repoint the telescope. If the point has set, mark that in our data structure and go back to 1

Since we always try to observe the point with the highest altitude we get the double benefit of scanning our object from top to bottom, and always attempting to take a point with at least half of its time in the sky still left. Scanning from top to bottom has the advantage of getting most of the points easily, and leaving the troublesome points for last. The highest points are up for 6 hours or longer, allowing easy observation runs, getting all the required spectra in one observation. Since we do all these first, we can work with most of our data very quickly, the last points being filled in at the end.

3.3. OBSERVING

The high velocity cloud we want to observe has the following characteristics:

- Coordinate Bounding Box: $(145°, 30°)$ to $(165°, 50°)$
- Maximum Signal Strength: $\approx 3K$
- Expected Velocity Range: $-180\,km/s$ to $-120\,km/s$
- Expected Doppler Shift from Local Hydrogen: $\approx +800\,kHz$

With such low signal strength, extended observation of a single point is needed. This necessitates the divide and conquer methodology of our tracking algorithm. Multiple separate observations per point allows us to minimize the amount of times that we lose a spectrum due to the object setting, and to point the telescope periodically.

To calibrate the final spectrum, we need to take two reference spectra - one with an artificial noise source of known intensity in our signal path, and one pointing to the object but observing in a slightly offset frequency range, so that we receive only noise. Both of these spectra are take initially, before we observe the 4 spectra of our main object.

All the spectra are generated on the fly by the telescope tracking program. We specify the amount of individual spectra we want, and the telescope automatically takes that amount of observations, calculates

the power spectrum, and creates an average spectra from the observed group. We take multiple observations per final spectrum to reduce the noise in each spectrum. For the spectrum containing artificial noise, we take 200 separate spectra and average them. For the offline spectrum (pointing at the object, recording a different frequency range) we take 400, and for the object itself we take 4 sets of 500, for the best signal to noise ratio possible in the time we have.

Since we take the DFT of each spectra, we can specify how many frequency channels we have in our final spectrum. We used a window of 1024 channels across the 5MHz bandwidth of our observations. This gives us a width of $5\,kHz$ per channel, which is more than small enough, since $8\,kHz$ wide H2 lines are common, and our distributed object will have significantly wider peaks than that, since the cloud has a wide range of velocities contained in it.

4. DATA ANALYSIS and IMAGE GENERATION

The data collection procedure took approximately 62 hours of telescope time, and a comparable amount of man-hours. At the end of the process, we had 89 spectra for 89 (l, b) points across our source. We now needed to extract information pertinent to our high velocity cloud from these spectra. Specifically, we want to reduce each spectrum to an intensity and velocity value of the cloud at that galactic position. To accomplish this we faced the challenges of extremely bad signal to noise ratio and very low intensity signals, for which we had to account using Fourier filtering and Gaussian fitting before attempting the correct transformations to produce an image of intensity, velocity and position of the cloud.

4.1. Spectrum Calibration and Fourier Filtering

The first step in getting any useful information from our spectra is the same as for our previous experiment in observing the hydrogen line - calibrating our spectra. The added twist in this experiment is the delicacy of our signal, which forces us to take extreme care to reduce the noise as much as possible. We accomplish this by using Fourier filtering during the calibration procedure.

To calibrate our spectra, we use the spectra we took that contains artificial noise and that looks at a slightly different frequency than our observation, henceforth referred to as the *S noise* and *S offline* spectra respectively, in comparison to the actual *S online* spectrum. We find the ratio of the online and offline spectra to smooth out the response characteristics of our feed and signal path, and use the following function to calibrate our spectrum:

$$ratio = \frac{s_{online}}{s_{offline}} \tag{13}$$

$$T_{cal} = 250K \tag{14}$$

$$T_{sys} = \frac{\sum S_{offline}}{\sum (S_{offine,calon} - S_{offline})} T_{cal} \tag{15}$$

$$s_{calibrated} = ratio * T_{sys} \tag{16}$$

In finding the ratio, we need to ensure that we do not increase the noise in the spectrum. The function of the ratio of these spectra is to smooth the feed, noise and signal path characteristics out of the observed spectrum. We will get the best results if we can divide our observed spectrum with a smooth line, characteristic of the setup without the hydrogen line observation in it. We achieve this by using Fourier Filtering on the offline spectrum. Fourier Filtering smooths out the noise spectrum into a line of the same shape but without the high frequency noise fluctuations. The IDL implementation is given in Appendix 7.2, and the effect is shown in Figure 7.

Fourier filtering works on the principle that the fine structure in frequency comes from long time delays in the autocorrelation function, which we take the Fourier Transform of. Thus, we truncate the transformed autocorrelation function above a preset time delay, and so remove the fine grain frequency fluctuations.

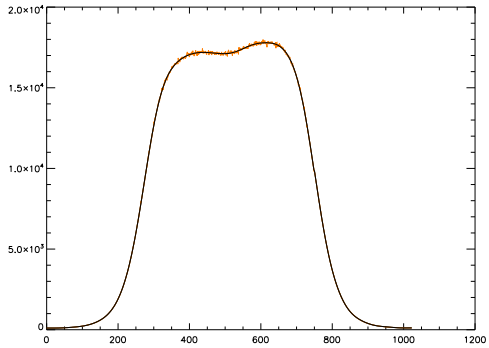

Fig. 7.— Fourier Filtering of a reference spectrum. In orange is the original spectrum, while the Fourier filtered plot is in black. Notice the smoothness of the filtered data.

After applying this to our reference spectra and calibrating all 89 of our observations using this method, we normalize it using a 3rd degree polyfit against the part of the spectrum we do not expect our signal to be. After this normalization we should be able to find something resembling our signal in our spectra. To illustrate the process, Figure 8 contains one of our spectra in various stages of processing. Notice the wide, low peak in signal strength between the two lines in the velocity (lowest) plot. We can say with fair certainty that this is our signal.

We took this data and built a data cube, with two axes representing Galactic Coordinates, and a third orthogonal axis representing velocity. We lines our spectra up in this cube, and wrote a program that moved through it 5 km/s chunks. Although we could see some structure around the velocity region we expect (-170 km/s to -120 km/s), we felt that our signals were not normalized quite correctly to really give us an idea of how the cloud changes, and that the velocities in the cloud is too wide for this method to give us much useful information. The fact that we could see some structure assured us that we are probably on the right track to tease this elusive signal out of the noise.

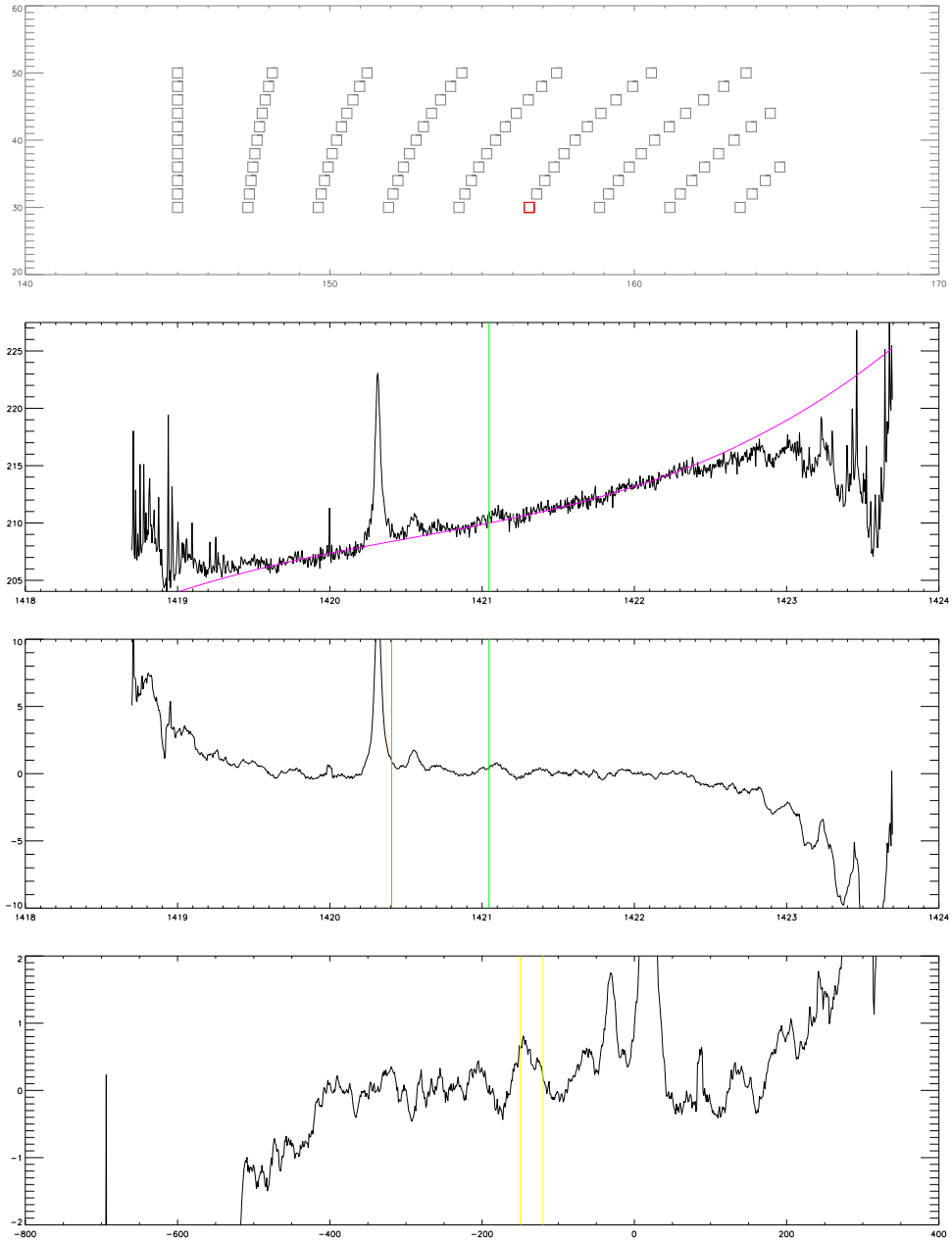

Fig. 8.— The process of calibrating our signal, including Fourier filtering, polyfitting, smoothing and predicting our signal. The topmost plot shows which point in Galactic Coordinates we are looking at. The second plot shows our polyfit to the noisy, calibrated data. The 3rd plot shows our smoothed, calibrated data, with a green line where we expect our signal and a red line where we expect local hydrogen. Finally, the lowest plot shows us where we expect our signal as calibrated against velocity. Notice the peak between the two yellow lines - this is almost certainly our signal.

4.2. Gaussian Fitting and Normalization

We now changed track and brainstormed about how we can reduce each spectrum to a velocity and intensity value that corresponds to our signal. We make the assumption that each spectrum contains mostly one small velocity range, and that the thermal dispersion through the cloud generates a spread-out spectrum. This spreading is characterized by a Gaussian distribution, thus we should be able to fit a Gaussian to what we believe to be our signal and extract a velocity as the center and intensity as the peak of the Gaussian fit. We applied this to each of our spectra by hand, and so reduced each spectrum to a velocity and intensity value. We sorted the spectra by position before we ran through this process, and plotted consecutive images as we fitted Gaussian to them. During the process we could see how our expected signal shifts around, and this helped to confirm that we are looking at a signal and not noise. There is no reason that noise would consistently have a width of about 60 km/s, and make small shifts through our expected velocity range, yet this is what we saw, thus we were confident that we detected our object. An example of a Gaussian Fit to what we believe is our signal can be seen in Figure 9. We normalized this further by finding the mean of each spectrum and centering the Gaussian and spectrum on that value. We then normalized that value to zero. The result of this was pushed through our image processing program, to generate the final images in Section 5.

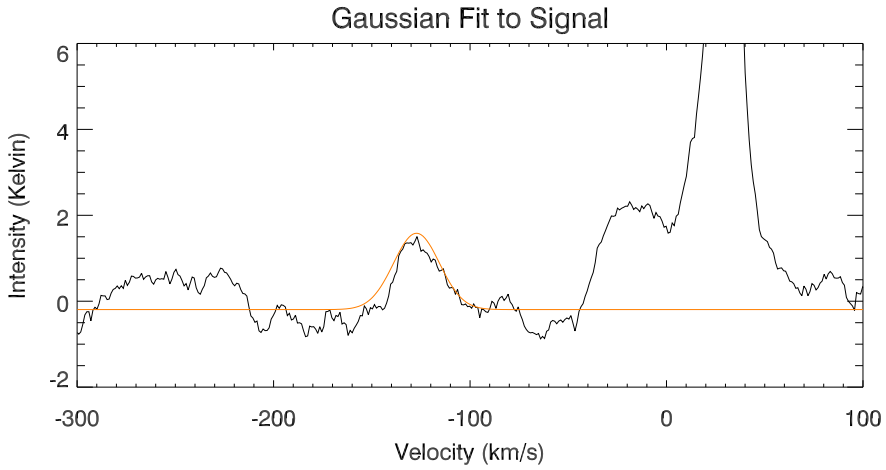

Fig. 9.— Our candidate signal, with a Gaussian fitted to it to find velocity and intensity.

5. FINAL IMAGES

We now present the images we created from our set of intensity and velocity values for each observed coordinate point. The image creation process used interpolation to generate a map of the sky at regular grid spacing from the irregular observation grid (due to the cosine factor in incrementing galactic longitude, Section 3.2.1). We generated various images by varying the amount of interpolation and magnification, until we were satisfied that we increased the pixel count by enough to form a reasonable image of the sky without creating artifacts or presenting misleading images. What is important to remember when looking at these images, is that any structure less that 2 angular degrees in size will not be resolved by our telescope at all, so even if we push the interpolation up a lot, as we did for Figure 11, we still only show the overall structure of the hydrogen distribution.

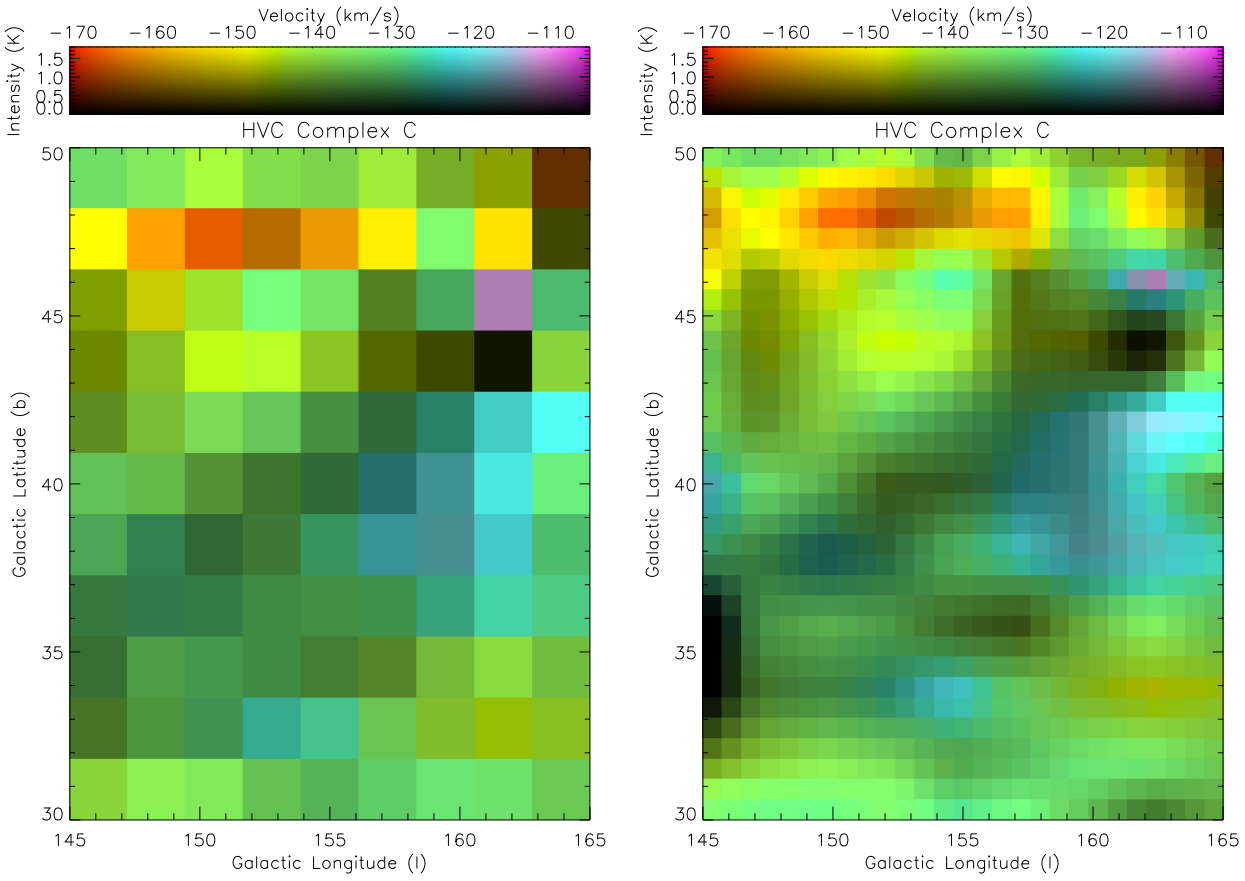

Fig. 10.— Our final plot combined velocity and intensity images. Color value represents velocity and color intensity represents intensity on a Kelvin scale. The left plot is our raw data, one observation per pixel. The right plot is a slightly interpolated image to generate smoother transitions between observation values.

Notice the structure revealed by our image. A hot, fast-moving part of the cloud sits in the top of our coordinate box, while it drops off to a diagonal area where the cloud thins out a lot.

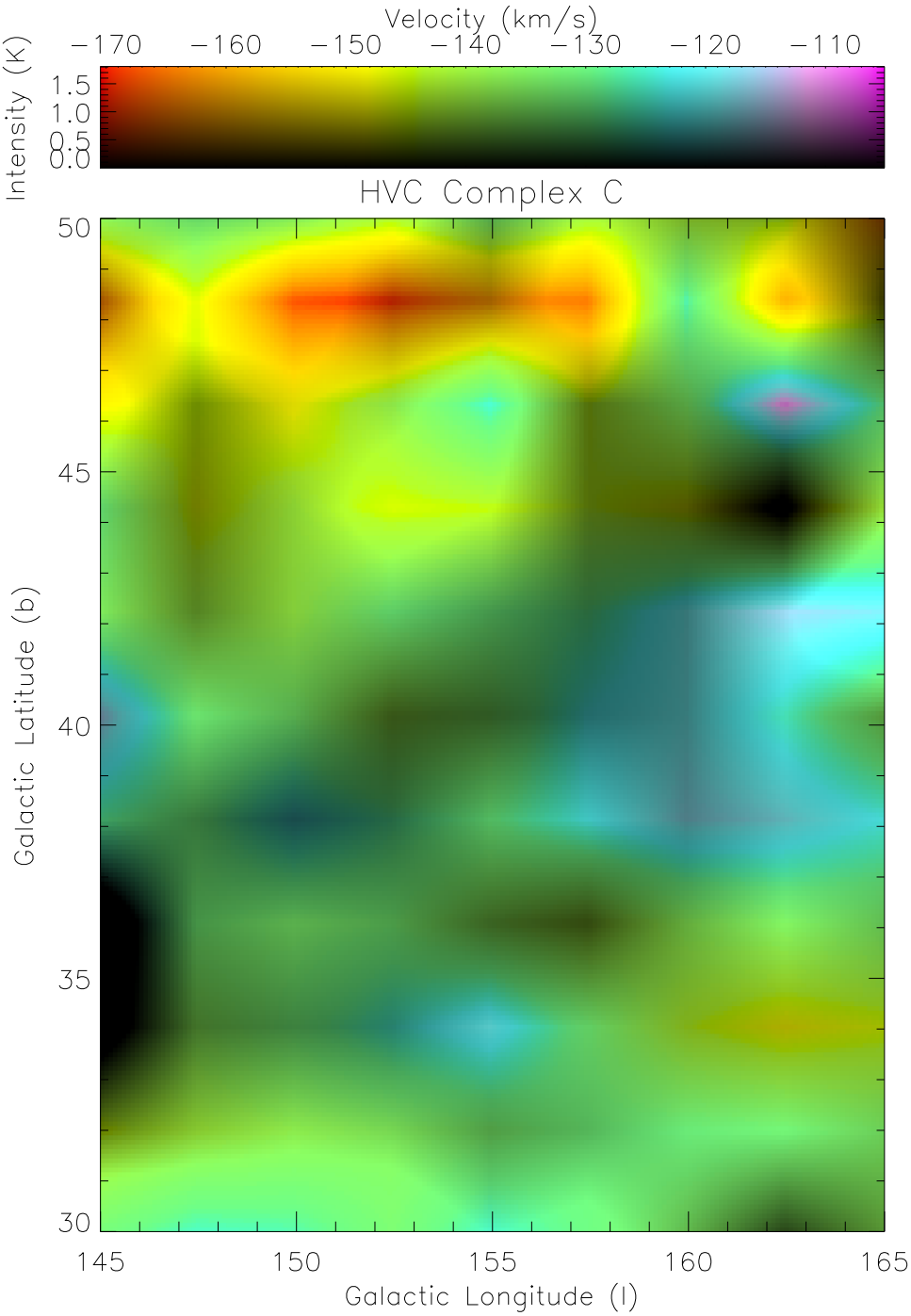

Fig. 11.— A highly interpolated image of our object, showing the large-scale structure of High Velocity Cloud Complex C.

6. CONCLUSION

Our final lab was a roaring success. We managed to observe an extended object of the sky with reasonable effectiveness, tracking our object and indexing our data through the process. We learned several methods to tease out a faint signal from the noise, including Fourier filtering, normalization and Gaussian fitting. We became adept at image creation, especially with data that does not map perfectly to a rectangular pixel grid. We learned the interpolation and image resizing that forms the tools in building a color image from our data. Our final images shows structure within the ranges we expect, and we confirm that we not only detected an extremely faint source, but managed to make a map of it. This was a great experience in radioastronomy, since we covered the whole process from picking objects to magazine-cover image. We would like to thank Josh Goldstone and Carl Heiles for their input in our project - it was a rewarding experience

7. APPENDICES

7.1. Tracking Algorithms and Coordinate Transforms in IDL

7.1.1. Galactic to Equatorial Coordinates

```
 ;+
 ; CONVERTS GALACTIC COORDINATES TO EQUATORIAL COORDINATES
 ;
 ; INPUTS
 ;    l       = Galactic Longitude
 ;    b       = Galactic Latitude
 ; KEYWORDS
 ;   radians  = If set, the values are accepted to be in radians, else it
 ;              will be assumed to be in degrees.
 ; OUTPUTS
 ;       returns in same format as input (radians iss /radians, else degrees)
 ;   struct.ra = ra
 ;   struct.dec = dec
 ;-
 FUNCTION LB2RD, l, b, radians=radians, precess=precess
   l = double(l)
   b = double(b)

  if (n_elements(radians) eq 0) then begin
    l_rad = l*!dtor
    b_rad = b*!dtor
  endif else begin
    l_rad = l
    b_rad = b
  endelse

  x = make_array(1, 3, /double)
  x[0] = cos(b_rad)*cos(l_rad)
  x[1] = cos(b_rad)*sin(l_rad)
  x[2] = sin(b_rad)

  R = make_array(3, 3, /double)
  R[0,0] = -0.054876
  R[0,1] = -0.873437
  R[0,2] = -0.483835
  R[1,0] = 0.494109
  R[1,1] = -0.444830
  R[1,2] = 0.746982
  R[2,0] = -0.867666
  R[2,1] = -0.198076
  R[2,2] = 0.455984

  xp = R##x

  y = make_array(2, 1, /double)
  y[0] = atan(xp[1],xp[0])
  y[1] = asin(xp[2])

  if (n_elements(radians) eq 0) then y = y*!radeg

  if keyword_set(precess) then precess, y[0], y[1], 2007, 2007.35
  return, {ra:y[0], dec:y[1]}

end
```

7.1.2. Equatorial to Horizontal Coordinates

```
;+
;RD2AA
;Converts declination, right ascension into azimuth/altitute.
;
; INPUT:
;       declination             in      Degrees or Radians
;       rightascencion  in      Degrees or Radians
;
; OUTPUT:
;       aa[0]   azimuth in Degrees
;       aa[1]   altitude in Degrees
;
; EXAMPLE INPUT:
;
; Generating Correct RA, Dec:
; precess, ra-in-deg, dec-in-deg, yearthen, yearnow
;
;
;-
FUNCTION RD2AA, declination, rightascension, RADIAN=radian

if not keyword_set(RADIAN) then begin
    ra_rad = rightascension*!dtor ;Convert to double precision if not already
    dec_rad = declination*!dtor
endif else begin
    ra_rad= double(rightascension)
    dec_rad= double(declination)
endelse

;Get the Hour Angle from the Right Ascension
lst_rad = ilst()*15*!dtor
ha_rad = lst_rad - ra_rad

;Make spherical coordinates into rectangular coordinates
x = make_array(1, 3, /double)
x[0] = cos(dec_rad)*cos(ha_rad)
x[1] = cos(dec_rad)*sin(ha_rad)
x[2] = sin(dec_rad)

;Make rectangular coordinates into spherical coordinates
station, lat, long
lat = lat*!dtor

hd_to_azalt = make_array (3, 3, /double, value=0.0)
hd_to_azalt[0,0] = -sin(lat)
hd_to_azalt[2,0] = cos(lat)
hd_to_azalt[1,1] = -1
hd_to_azalt[0,2] = cos(lat)
hd_to_azalt[2,2] = sin(lat)

;Multiply time!
xp = hd_to_azalt##x

aa = make_array(2, /double)
aa[0] = atan(xp(1),xp(0))/!dtor
aa[1] = asin(xp(2))/!dtor

return, aa

END
```

7.2. Fourier Filtering a Spectrum

```
;+
; NAME
;  Fourier Filter
;
; PROCEDURE
;  ffilter, spectrum, tau_max=tau_max
;
; MODIFICATIONS
;  4/24 - created during lecture by the VOGONS
;   Niels Joubert, Patrick Lii, Samantha Thompson, Brent Macombe, John Zhang
;
;-
function ffilter, spectrum, tau_max=tau_max, noplot=noplot

; generate inverted arrays
nonoiseinv = dblarr(2*n_elements(spectrum))
nonoiseinv[1: n_elements(spectrum)] = reverse(spectrum)
nonoiseinv[n_elements(spectrum)+1: *] = spectrum[1:*]
nonoiseshift = shift(nonoiseinv, n_elements(spectrum))

; take inverse fourier transform
ff = fft(nonoiseshift, /inverse)

; set tau_max
if not keyword_set(tau_max) then tau_max = 45
ff[tau_max: n_elements(ff)-tau_max] = 0

; take fourier transform
ffinal = fft(ff)

if not keyword_set(noplot) then begin

; plot spectra
    window, 0
    plot, ffinal[0:n_elements(spectrum)-1]
    oplot, spectrum, color=!orange

; plot spectrum-ffinal
    window, 1
    plot, spectrum-ffinal

endif

return, ffinal

end
```