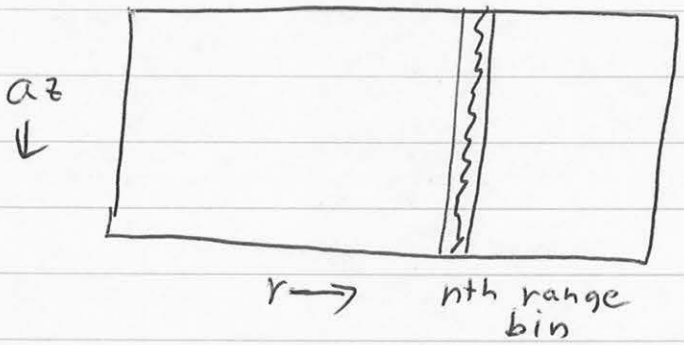
Range Migration

We have developed the SAR processing algorithm by devising a matched filter for the phase history of a scattering object. The phase history was related to the range history by a scalar relation:

$$\phi(t) = \frac{-4\pi}{\lambda} r(t)$$

Implicit in this is the notion that range changes as a function of time.

We also isolate the return from objects of a given range by implementing the matched filter along lines of constant range, or range "bins", in the range compressed image:



The "width" of the range bin is the range spacing, or approximately the range resolution of the system.

Now, what happens if the range history above exceeds the width of the range bins? Clearly the energy from a given object would be spread over several bins, therefore the focus from our algorithm, which is a single-column matched filter, would not be complete.

We call this apparent movement of the energy from one object through several range bins <u>range migration</u>.

Magnitude of range migration

Recall our equation for range history:

$$r^2(t) = r_{DC}^2 + v^2 t'^2 + 2v^2 t_{DC} t'$$

↑ quadratic term ↑ linear term from Doppler centroid

Just as in the case for phase, the range has both a linear and a quadratic part - the linear term goes to zero for non-squinted geometries ($t_{DC} = 0$).

This variation in range is called migration if the magnitude is greater than the range bin width. Sometimes the quadratic term is called range curvature and the linear term range walk.

Once again apply our square root approximation:

$$r(t) \doteq r_{DC}\left(1 + \frac{1}{2}\frac{v^2 t'^2}{r_{DC}^2} + \frac{v^2 t_{DC} t'}{r_{DC}^2}\right)$$

$$= r_{DC} + \frac{1}{2}\frac{v^2 t'^2}{r_{DC}} + \frac{v^2 t_{DC} t'}{r_{DC}}$$

Since the constant term does not change with time, we'll ignore it for now. Let's estimate the magnitude of the other two terms compared to a range bin. We'll suppose the complex sample rate in range is f_s, corresponding to a time per sample of $1/f_s$. Then the spacing in range is

$$\text{spacing} = \frac{c}{2 f_s} = \Delta_r$$

Hence the 2nd order term in range bins is

$$\frac{\frac{1}{2}\frac{v^2 t'^2}{r_{DC}}}{\frac{c}{2f_s}} = \frac{f_s v^2 t'^2}{c \cdot r_{DC}}$$

and the linear term is

$$\frac{\frac{v^2 t_{DC} t'}{r_{DC}}}{\frac{c}{2f_s}} = \frac{2 f_s v^2 t_{DC} t'}{c\, r_{DC}}$$

Actually we'll find it simpler to first evaluate the range spacing Δ_r and think of that as units for the migration. Then the total migration is just

$$\text{migration} = \frac{v^2 t'^2}{2 r_{DC} \Delta} + \frac{v^2 t_{DC} t'}{r_{DC} \Delta}$$

Clearly the migration will be larger for longer integration times and also for larger squint angles. The number of bins affected will be inversely proportional to the range spacing (equal to resolution δ_r for critical sampling).

Examples:

ERS-1: First, we determine the integration time. The azimuth beamwidth is

$$\frac{r\lambda}{\ell}$$

so the integration time is

$$\frac{r\lambda}{v\ell} = \frac{850\times10^{3} \times 5.66\times10^{-2}}{7550 \times 10}$$

$$= 0.64 \text{ s}$$

Thus t' varies as $\pm$ 0.32 second. The range spacing is set by the 18.96 MHz sample rate:

$$\Delta_r = \frac{c}{2 \cdot 18.96 \text{ MHz}}$$

$$= 7.91 \text{ m}$$

Hence the maximum migration from the 2nd order term is

$$\frac{v^2 t'^2}{2 r_{DC} \Delta} = \frac{(7550)^2 (0.32)^2}{2 \cdot 850\times10^{3} \cdot 7.91}$$

$$= 0.434 \quad \leftarrow \text{less than one bin max } (\pm 0.434)$$

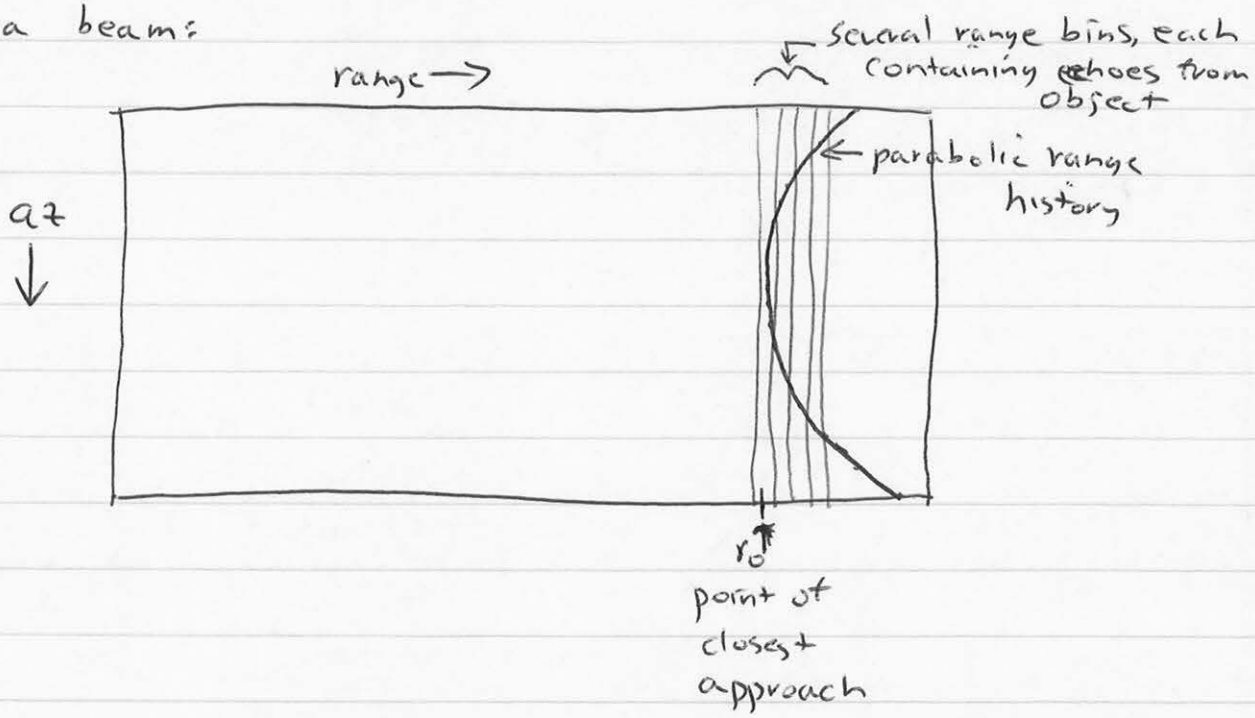
But the linear term depends on Doppler centroid. Suppose we are squinted forward by one antenna beamwidth so that $t_{DC} = 0.64$ s. Then the linear term will be

$$\frac{v^2 t_{DC} t'}{r_{DC} \Delta} = \frac{(7550)^2 \cdot 0.64 \cdot 0.32}{850\times10^{3} \cdot 7.91}$$

$$= 1.736 \text{ bins} \quad \leftarrow \text{here we have to start worrying}$$

So for this example we migrate over a couple of bins, mainly due to the linear term.

Let's try to visualize the migration in our picture of the range compressed data. We do this by plotting the path taken by a scattering point as it moves through our antenna beam:



What happens when we apply our SAR algorithm to these data? The object will appear in several range bins in the output, and at reduced azimuth resolution in each. (Why is that? Explain several different ways).

Thus the point will be blurred in both range and azimuth. That is, it won't focus to one pixel.

But, if the migration is small such as in the ERS case, the blurring won't be too severe, and perhaps to give up a factor of two in resolution won't be too bad. But for most cases we aren't so willing to give up performance. Take another example, the space shuttle SIR-C radar:

In this system $r_{DC} \approx 300$ km, $\lambda = 0.24$ m for the L-band channel, and the sample rate is 45 MHz:

$$\Delta_r = 3.33 \text{ m}$$

$$t' = \pm 0.4 \text{ s}$$

The 2nd order migration is

$$\frac{v^2 t'^2}{2 r_{DC} \Delta} = 4.5 \text{ pixels (or bins)}$$

and for a one-antenna squint

$$\frac{v^2 f_{DC} t'}{r_{DC} \Delta} = 18 \text{ bins} \quad \text{(and this is } \pm 18!\text{)}$$

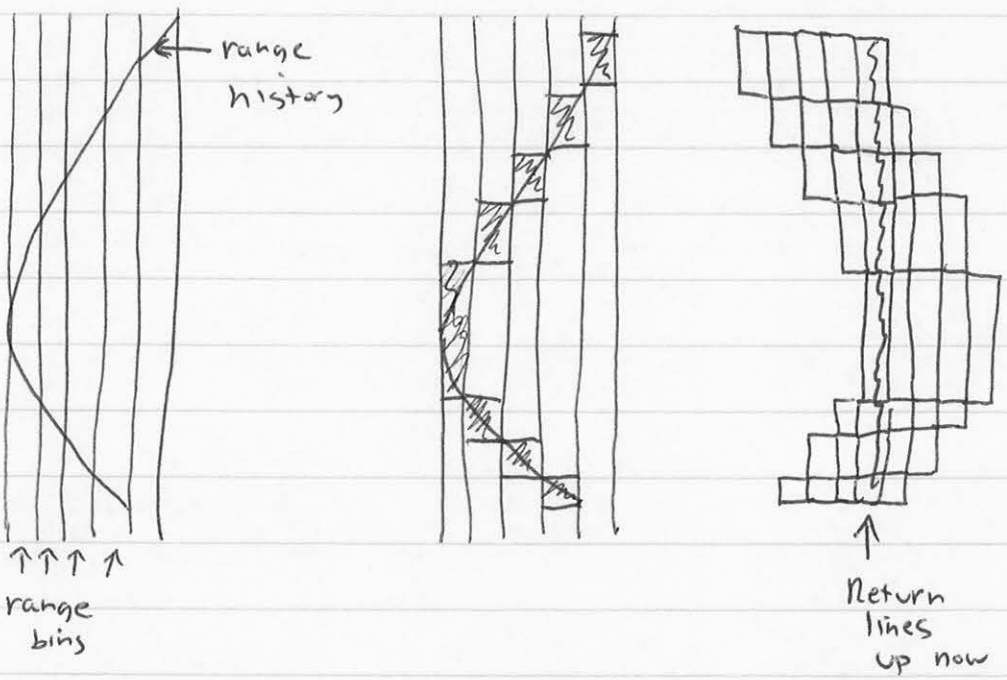
So a very large reduction in resolution would occur. How can we accommodate this so we can generate high quality images?

Range migration algorithms

One way to compensate for the migration would be to define a two-dimensional reference function, and correlate the range compressed data in two dimensions rather than one. This would result in a fully focused image.

But, replacing all of our azimuth 1-1) correlations with 2-1) correlations would be slow indeed on our computer, perhaps 100's of times slower than the 1-1) calculations.

Fortunately, another, more economical approach is possible. It involves predistorting the range compressed data so that our initial column-compression algorithm will work. Lets look at a detail picture of the r.c. data to see how this might work:



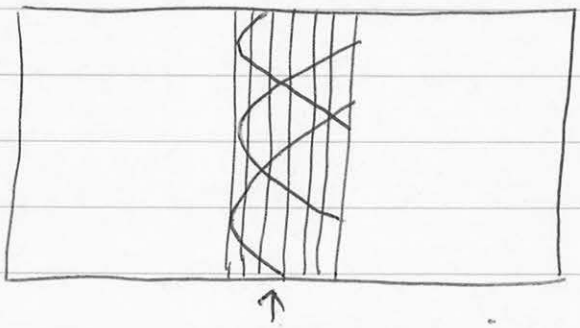
Initial Range History | Bins with signal identified | Distorted data matrix

By simply shifting the rows of the matrix back and forth we can align the echo from a given reflector in a single column, thus we can read the data out from that column and compress it with our usual SAR matched filter. This is much more computationally efficient than a 2-D filter.

Because this type of algorithm involves "cutting" pieces of the range bin data and "pasting" it into a new column, it is often called <u>cut and paste</u> range migration correction.

Variants of time-domain cut and paste are used all the time, not only in radar image processing, but in seismic data processing where the data are said to "be migrated".

But there is one thing wrong with our algorithm as presented. What if there are two objects at the same range but displaced in azimuth? For an image of course there are many objects at the same range. We have this situation:



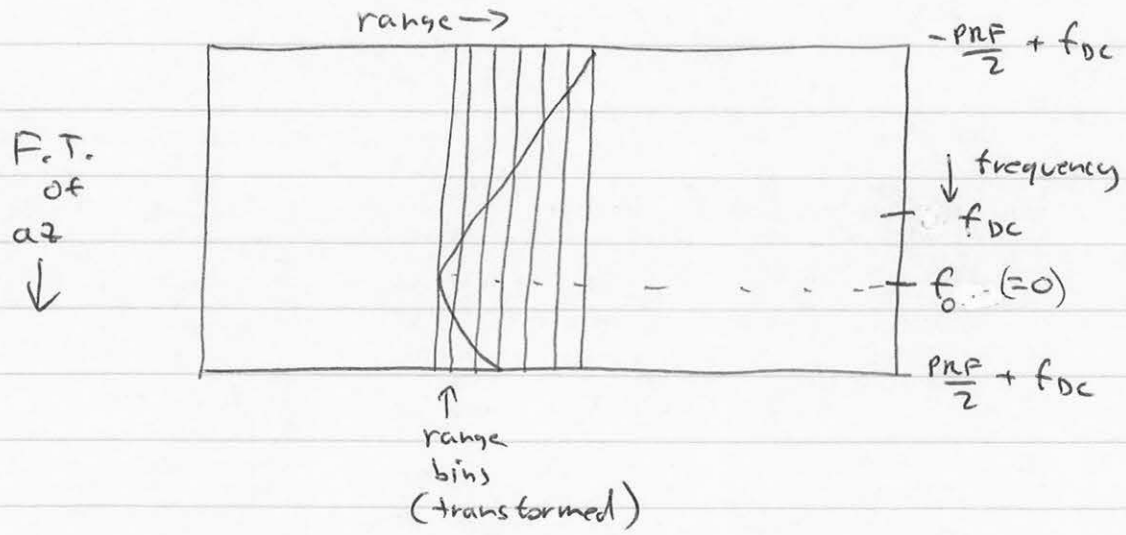
three range histories with same r_0!

How can we avoid having to regenerate all of our cut and paste points for every azimuth location (bin) in our desired range bin column?

Frequency-domain migration

Fortunately we have a way around this. While the range (and phase) histories of the echoes are offset by different amounts in time, in the frequency domain all points at the same range go through identical frequency vs. time histories. In other words, all points start at the same Doppler frequency, decrease at the same rate, and exit at the same Doppler. Also, because of the near quadratic nature of the azimuth chirp there is a nearly linear frequency to time mapping.

Suppose we examine the range compressed data after it has all been Fourier transform, and identify the frequency history of a scatterer:



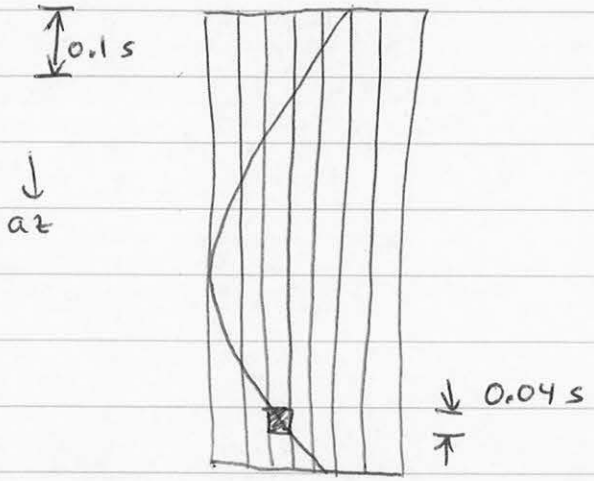
So we have applied a full set of column-wise Fourier transforms to the range compressed data. We can now apply our cut and paste approach in the frequency domain on a column by column basis, as before we need to update our cut points and f_{x0} f_R parameters on a range bin by range bin basis, but this isn't too complicated.

So our migration processor has the following steps:

1) Range compress each line in range
2) Transform the entire array in azimuth
3) Loop over range bins:
   - 3a) Apply cut and paste to get proper bins
   - 3b) Multiply column by conjugate of reference transform
   - 3c) Inverse transform columns
4) Write out result

Interpolation

Cut and paste yields reasonable images if the number of cut points is small and each segment is long enough to be meaningful. Quantitatively this means that the time-bandwidth product per segment be significantly greater than 1. Look at this situation:



Suppose our integration time is 0.7s as above and the Doppler rate is 100 Hz / s. Although the total bandwidth is 70 Hz for a TBP (time-bandwidth product) of 49, for the indicated cut and paste segment the time is only 0.04s and TBP is

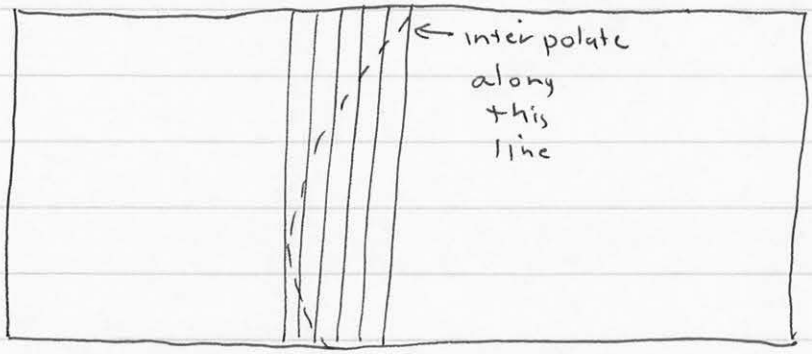
$$TBP = 0.04 \times (0.04 \times 100)$$
$$= 0.16$$

Thus we wouldn't expect much compression from this signal, it is too "spread out" in the frequency domain. Cutting and pasting thus retrieves only a fraction of the echo's signal.

Another problem with cut and paste techniques is that the sharp transitions in the frequency domain cause "ringing" or sidelobes

in the time domain of the compressed signal. Clearly this effect gets worse with more cuts also.

Our way around both of these limitations is to use an interpolation method rather than cut and paste to predistort the azimuth frequency domain signal. Once again we must use the full 2-D matrix of transformed, range compressed, signals. Instead of selecting sections from each column, though, we interpolate the path followed by the range history:

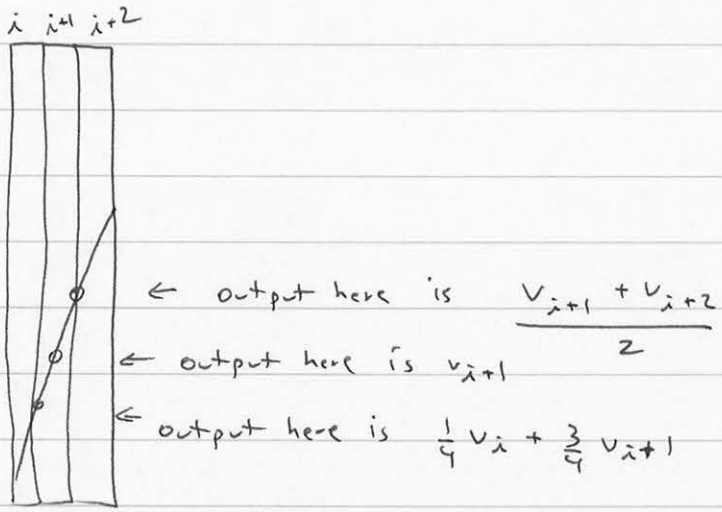


Interpolation along an arbitrary path in a 2-D matrix is a straight-forward, but tedious task. It is however, much easier than deriving the full 2-D reference function for each range bin. Many texts exist on the subject, and its beyond our scope to review them.

But, we need to know enough to choose the proper form of an interpolation kernel. We noted that the energy can be spread out over several bins because of low time-bandwidth relations, so the "wider" an interpolation kernel we choose, the more accurately we recover the echo.

The width is important because more points implies more computations.

Let's just assume that we want several points in our kernel, say about 8. Compare this to a simple linear interpolator:



Linear interpolator

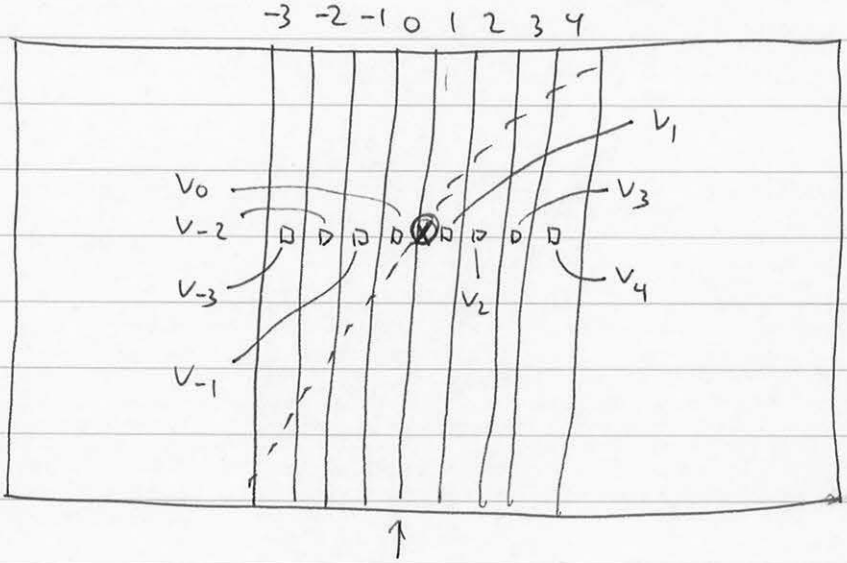
The linear interpolator simply estimates the position of the history to non-integer range bin locations, and calculates a weighted sum for the interpolated result. While often better than cut and paste, linear interpolators generate a lot of high-frequency noise.

So, we will use a multipoint sinc interpolator, apt for band limited signals. We'll choose an eight-point interpolator as a trade off between accuracy and computational time. If the range bin index is i, the output of this interpolator is

$$V_{out} = \sum_{i=-3}^{4} V_i \, \mathrm{sinc}(f + i)$$

where the desired track lies between points $i=0$ and $i=1$, and f is the fractional distance from $i=0$ to $i=1$. The limits may be extended at the cost of more computer time.

Let's quickly see this visually:



desired point
at a $(0<a<1)$

$$V_{out} = V_{-3}\,\mathrm{sinc}(a-3) + V_{-2}\,\mathrm{sinc}(a-2) + V_{-1}\,\mathrm{sinc}(a-1)$$

$$+ V_0\,\mathrm{sinc}(a) + V_1\,\mathrm{sinc}(a+1) + V_2\,\mathrm{sinc}(a+2)$$

$$+ V_3\,\mathrm{sinc}(a+3) + V_4\,\mathrm{sinc}(a+4)$$