Azimuth Compression: A Conceptual Overview

Elizabeth Wig

10/25/2023

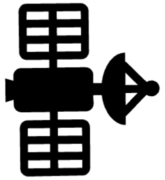
The types of azimuth compression

- What are the types of azimuth compression?
- Why does the resolution get better with each?

- To answer this, we answer a fundamental question: what does our radar see as it sends out a series of pulses over time?

- Real aperture radar
- Unfocused synthetic aperture radar
- Focused synthetic aperture radar

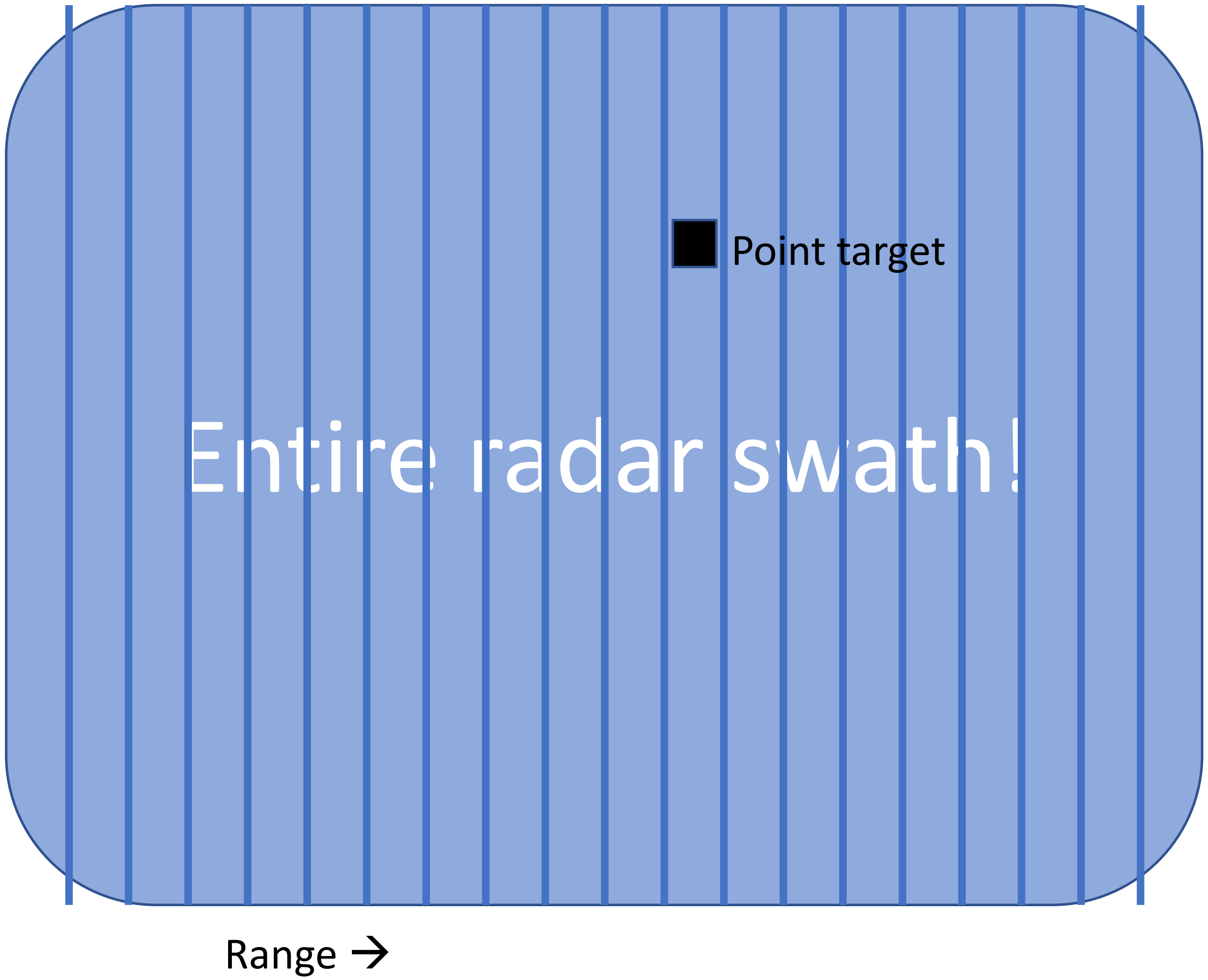
Assume we have a range-compressed swath



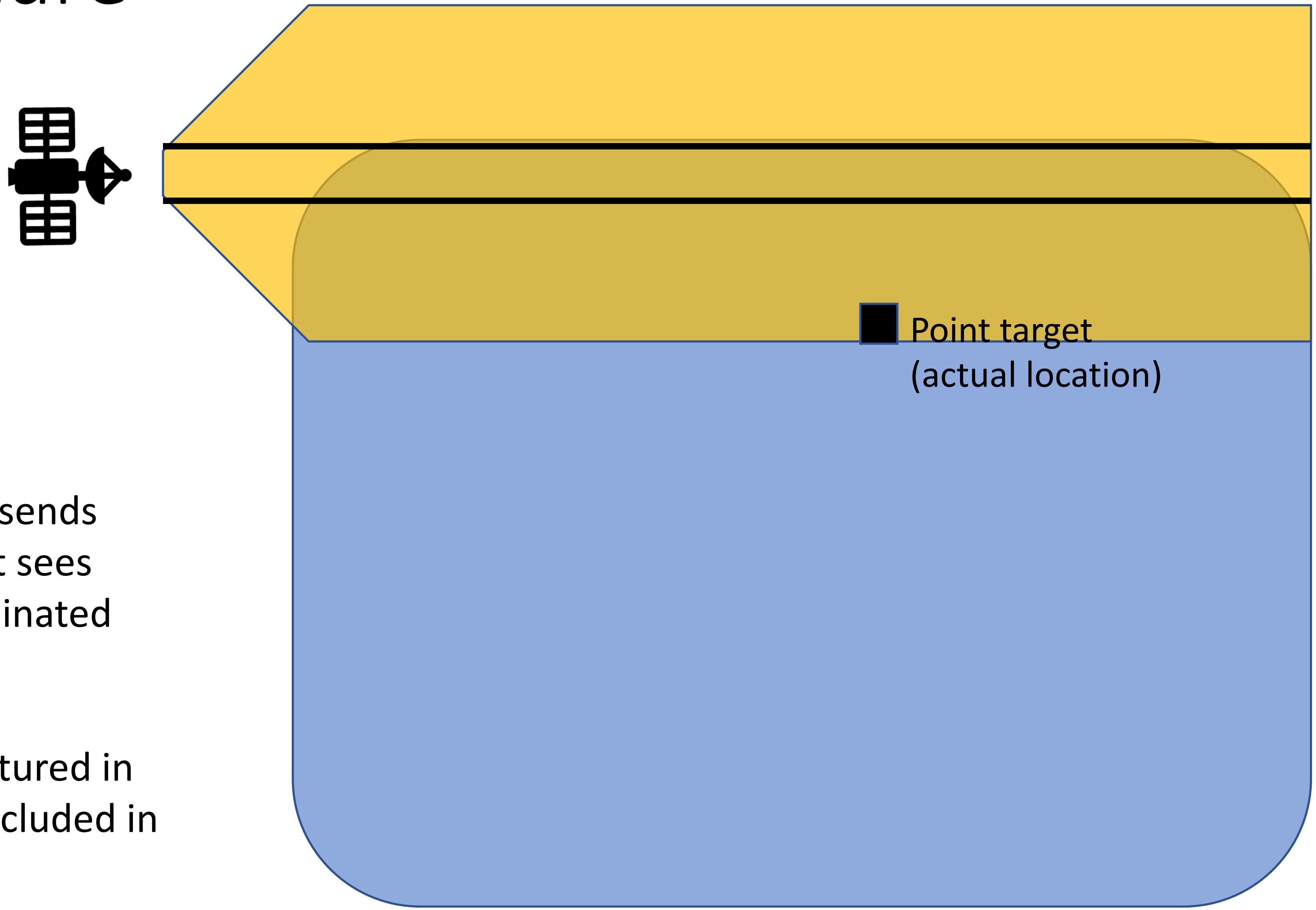
Things are well-focused within each range bin (horizontal) but not along the azimuth direction (vertical).

Our image probably contains a bunch of vertical streaks.

This point target will look “smeared” through the range bin. Why?



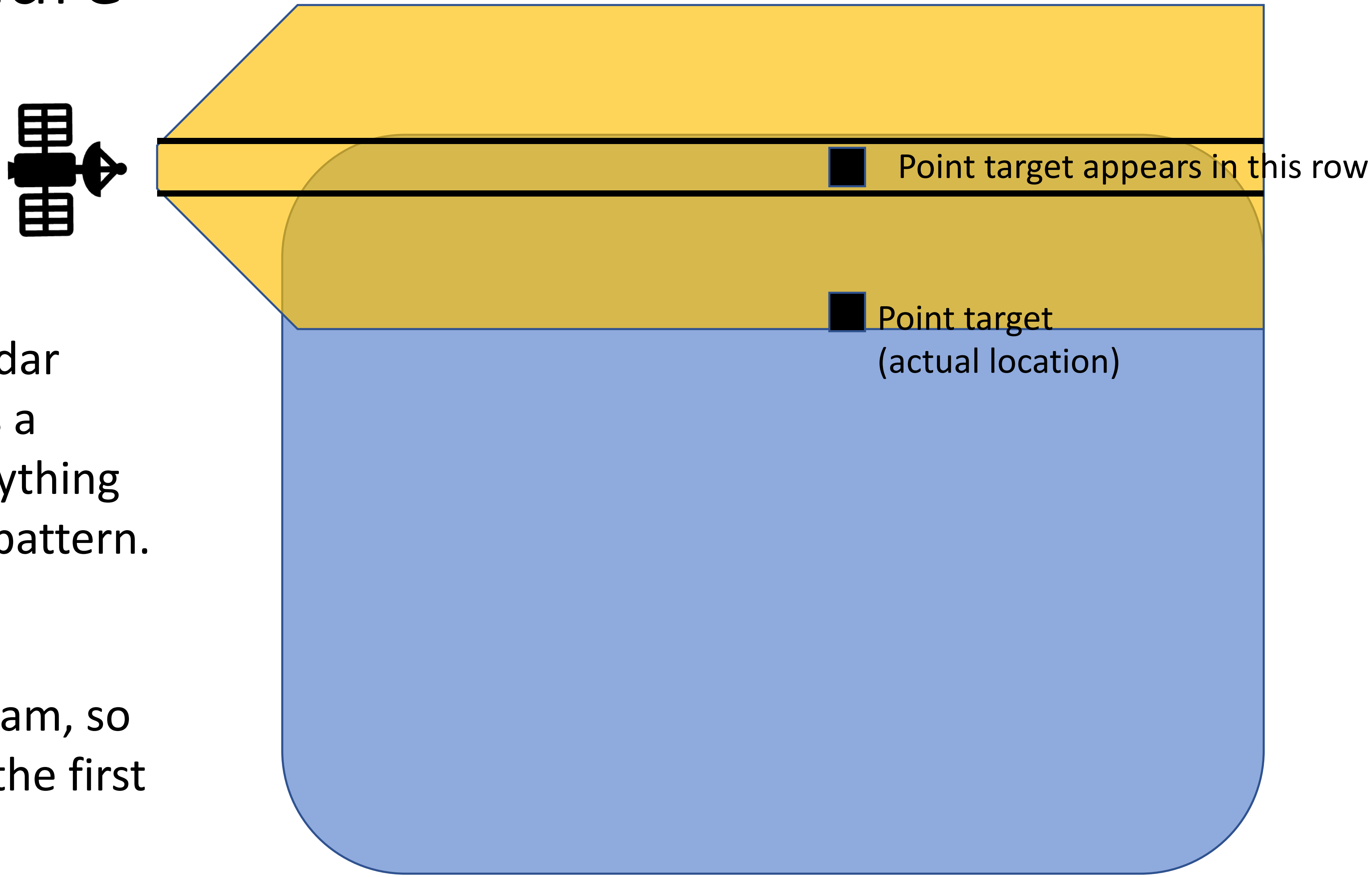
Real Aperture Radar



At time t=0, our radar sends and receives a pulse. It sees everything in the illuminated pattern.

The point target is captured in this beam, so it gets included in the first azimuth row.

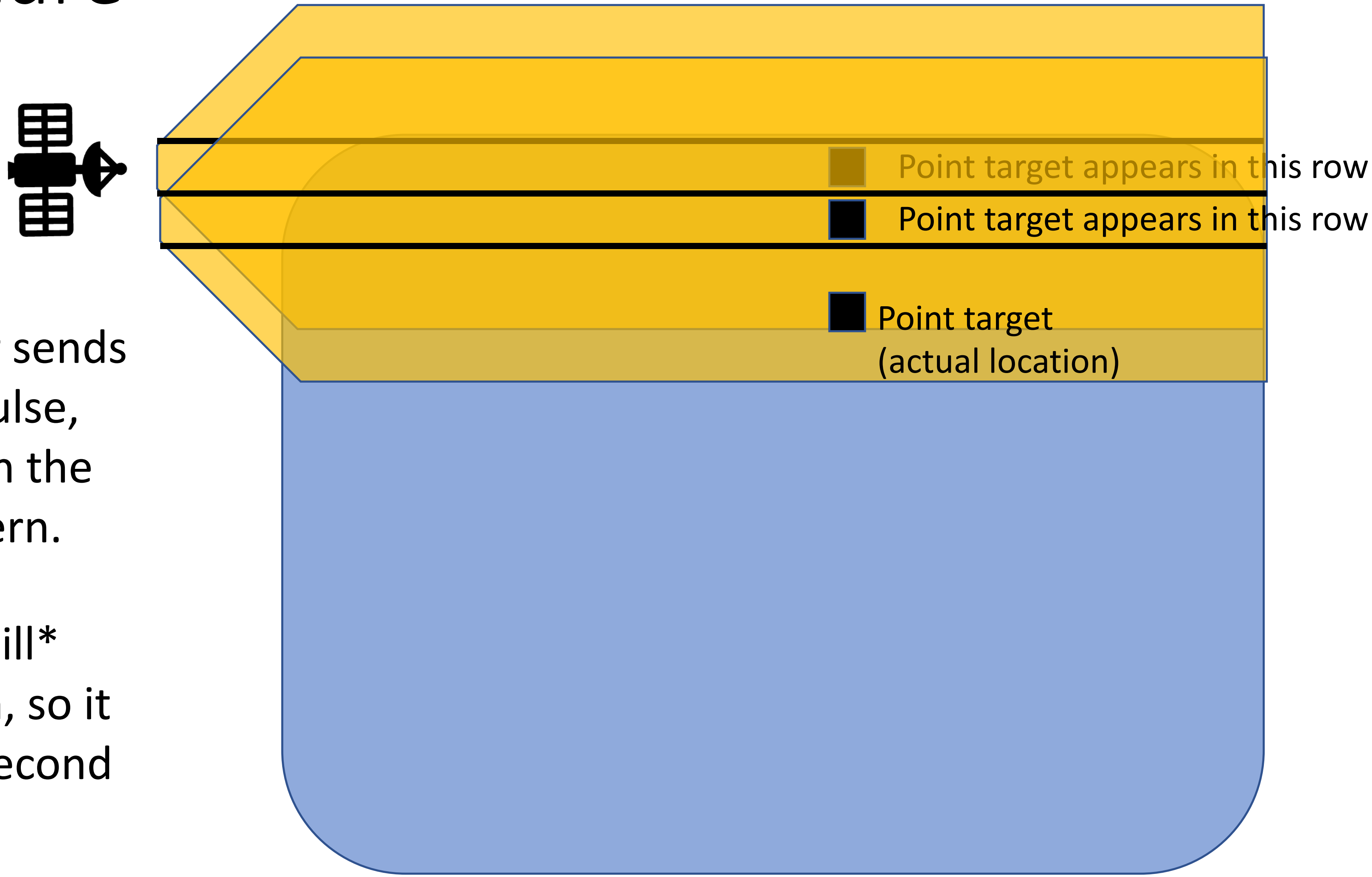
Real Aperture Radar



At time t=1, our radar sends and receives a pulse. It sees everything in the illuminated pattern.

The point target is captured in this beam, so it gets included in the first azimuth row.

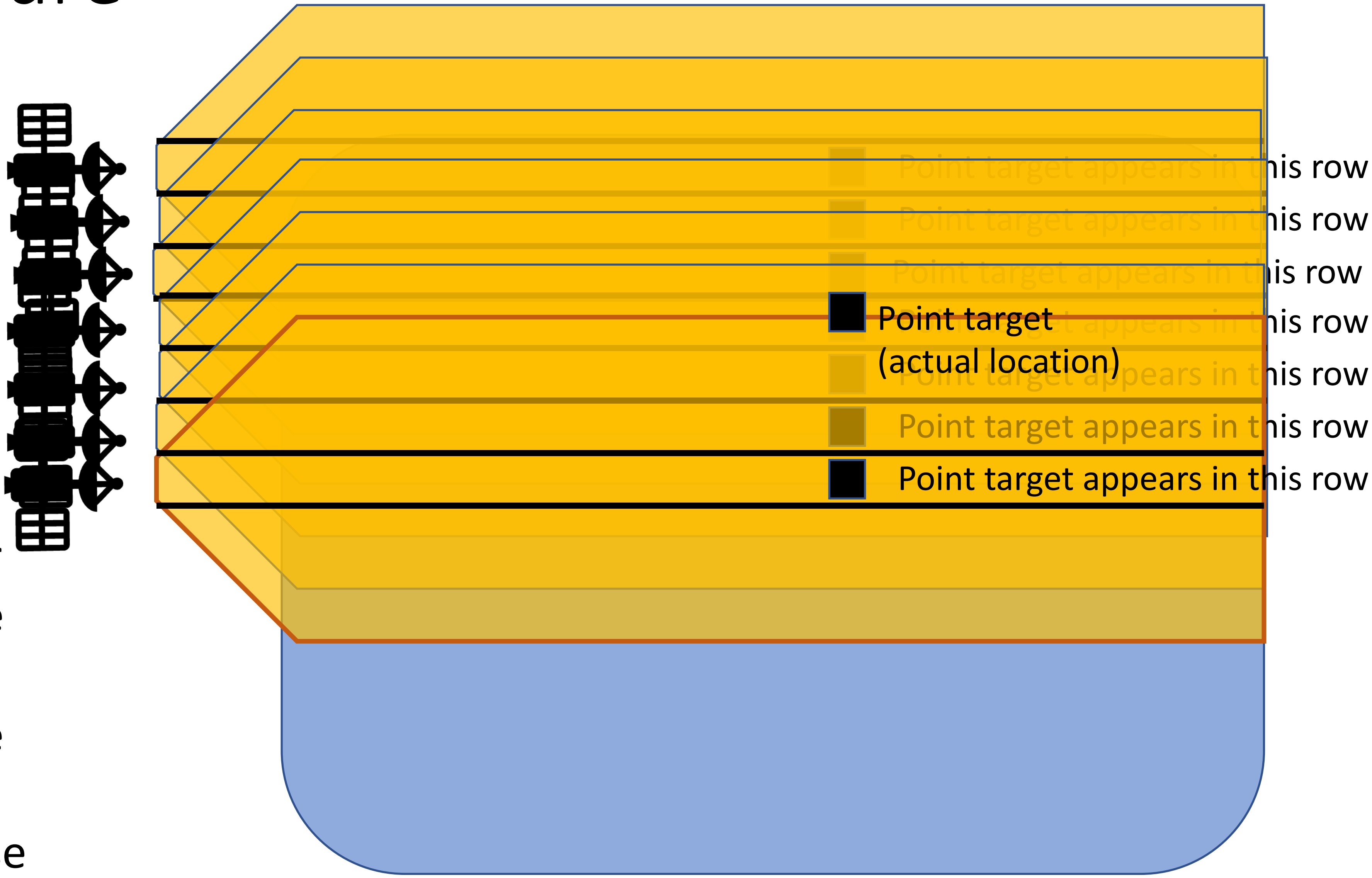
Real Aperture Radar



At time t=2, our radar sends and receives a new pulse, and sees everything in the new illuminated pattern.

The point target is *still* captured in this beam, so it gets included in the second azimuth row of data.

Real Aperture Radar

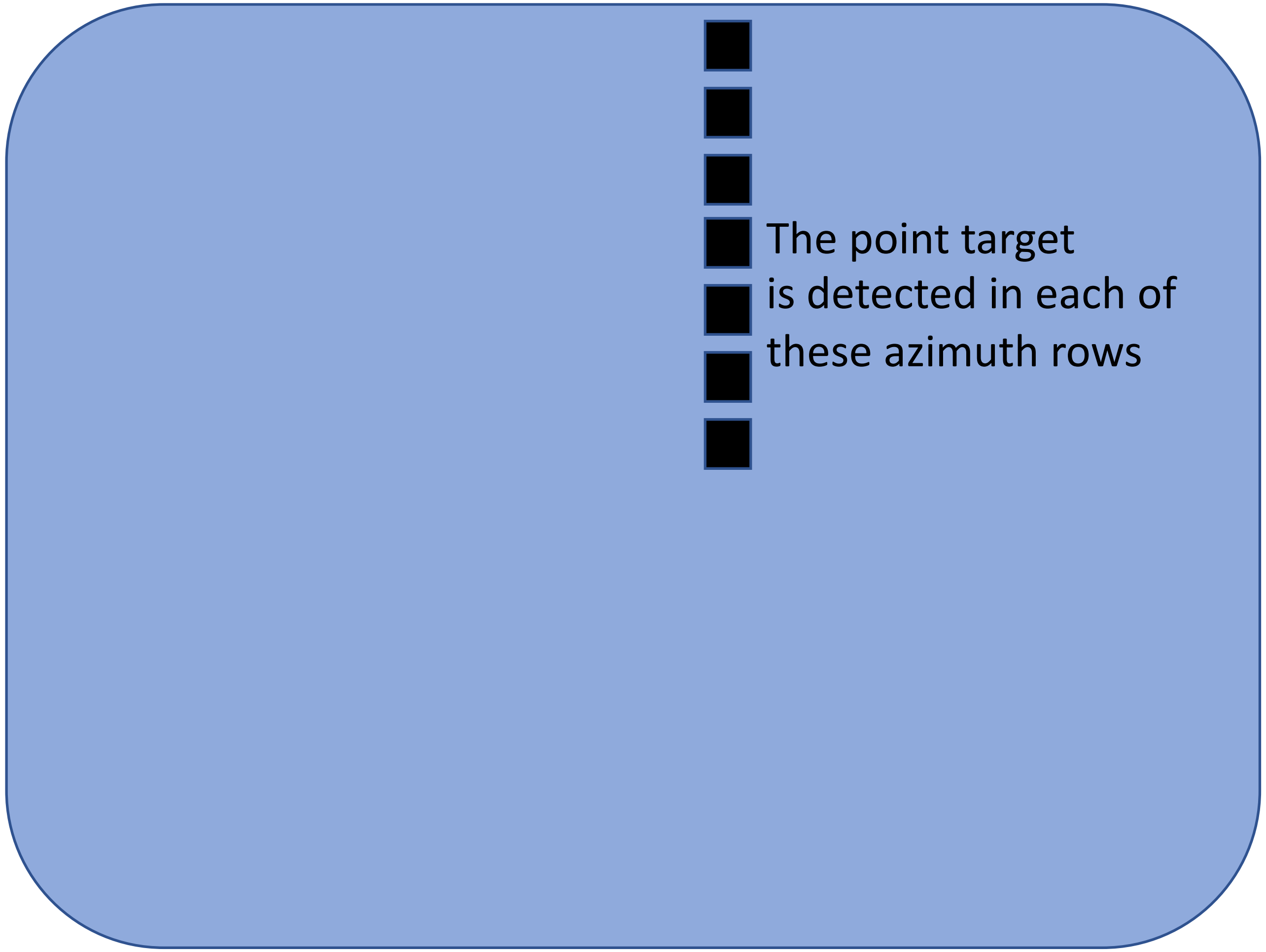


There are a lot of locations as the radar flies, where it can see the point target in response to the pulse it sends. The target is included in **all** of these azimuth rows.

Real Aperture Radar

There are a lot of locations as the radar flies, where it can see the point target. The target is included in **all** of these azimuth rows.

As a result, this point target gets “smeared” across all the rows where the radar can perceive it.



How do we fix this?

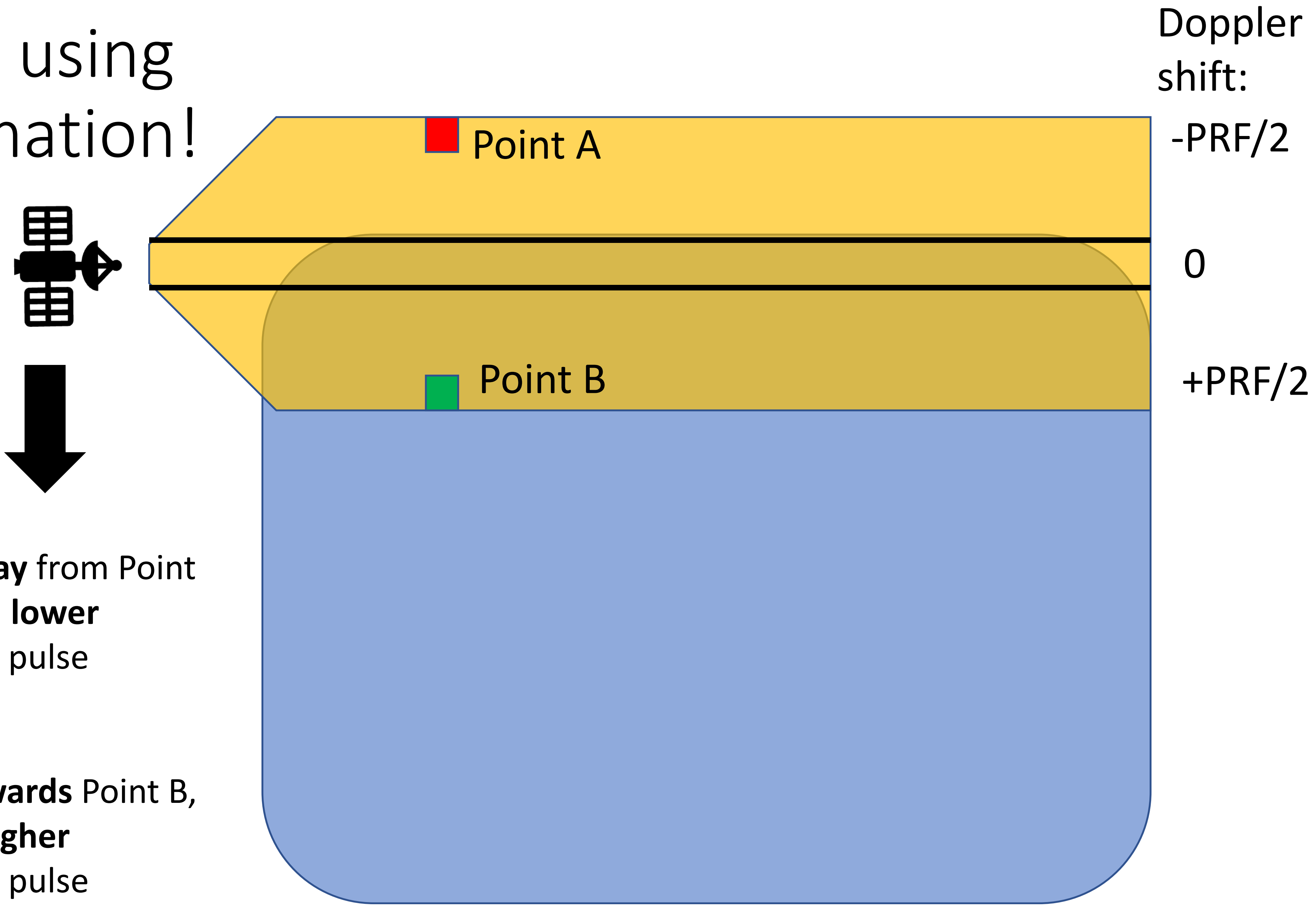
How do we fix this egregious smearing?

Using Doppler information!

Time for **unfocused SAR**!

We can fix this using Doppler information!

Our platform is moving this way

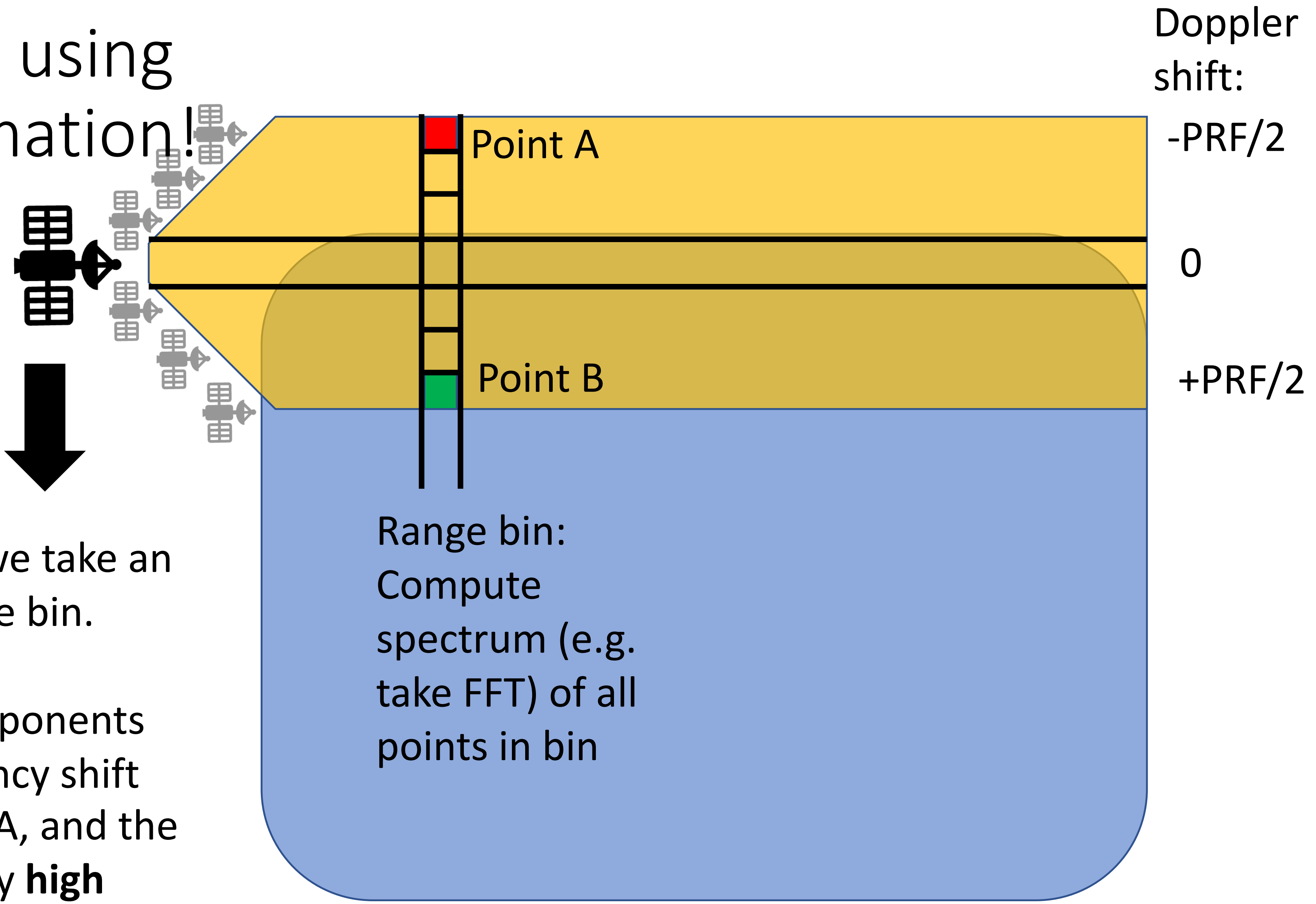


Our platform is moving **away** from Point A, so we expect it to have a **lower** frequency-shifted returned pulse relative to other points

Our platform is moving **towards** Point B, so we expect it to have a **higher** frequency-shifted returned pulse relative to other points

We can fix this using Doppler information!

How to find what the frequency shift is within this pulse series? Take a Fourier Transform!

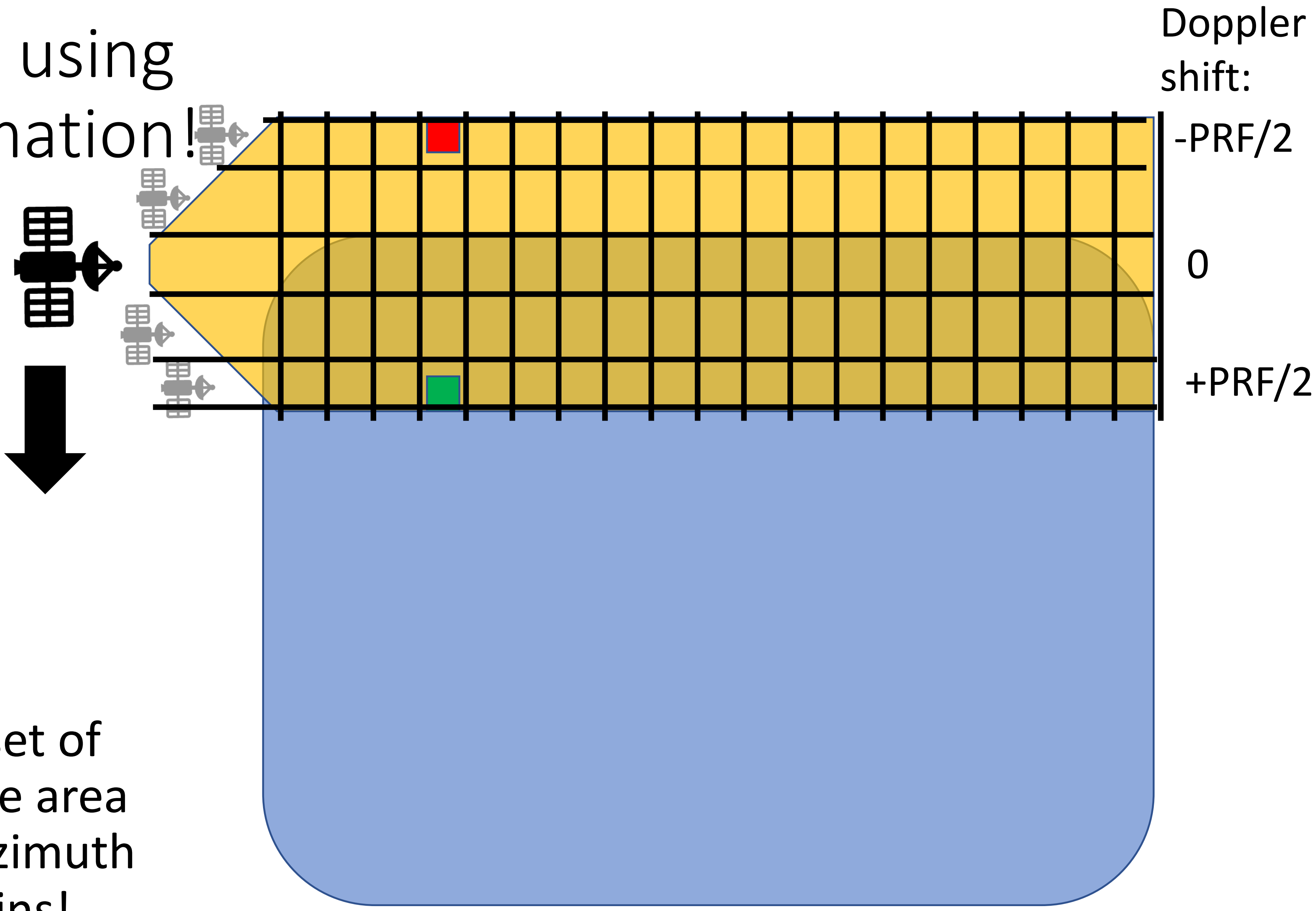


For each pulse series, we take an FFT across a given range bin.

This “sorts” all the components with a very **low** frequency shift into the bin with Point A, and the components with a very **high** shift into the bin with point B

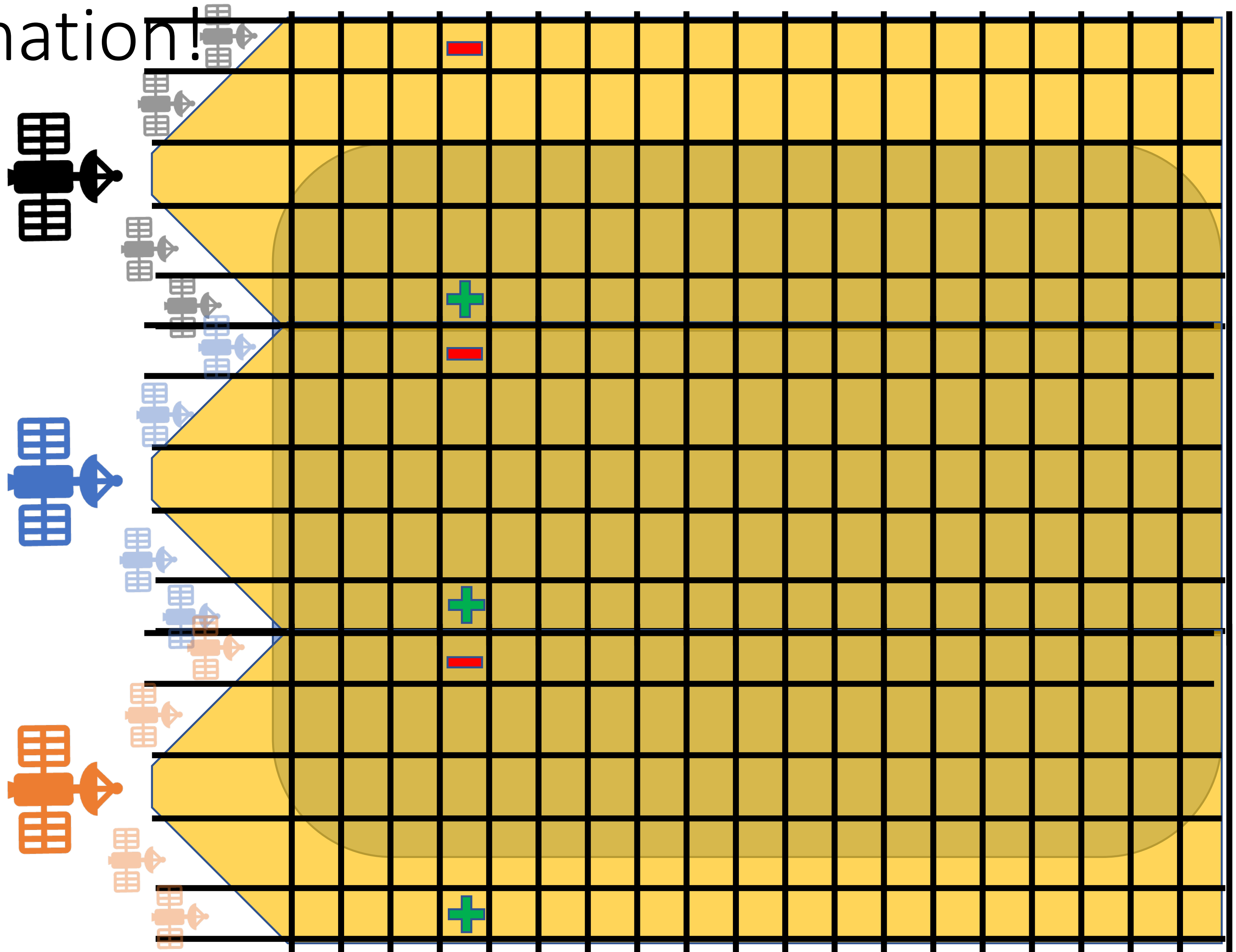
We can fix this using Doppler information!

Now, instead of only filling in one row of information, we can actually sort **everything** we detect in this burst into the bins they belong in.



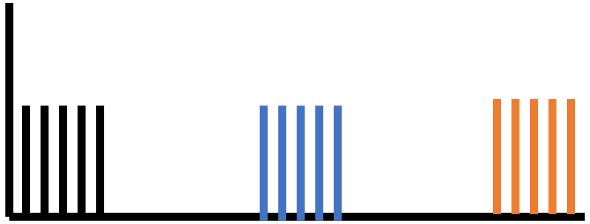
Imaging from a subset of pulses over the same area can fill in multiple azimuth rows, for all range bins!

We can fix this using Doppler information!



We could fill in the whole image, then, from just a few **sets of pulses (bursts)** over totally unique areas.

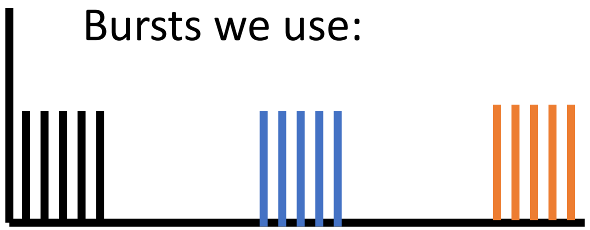
The targets will look much less “smeared,” because we can put them in Doppler bins.

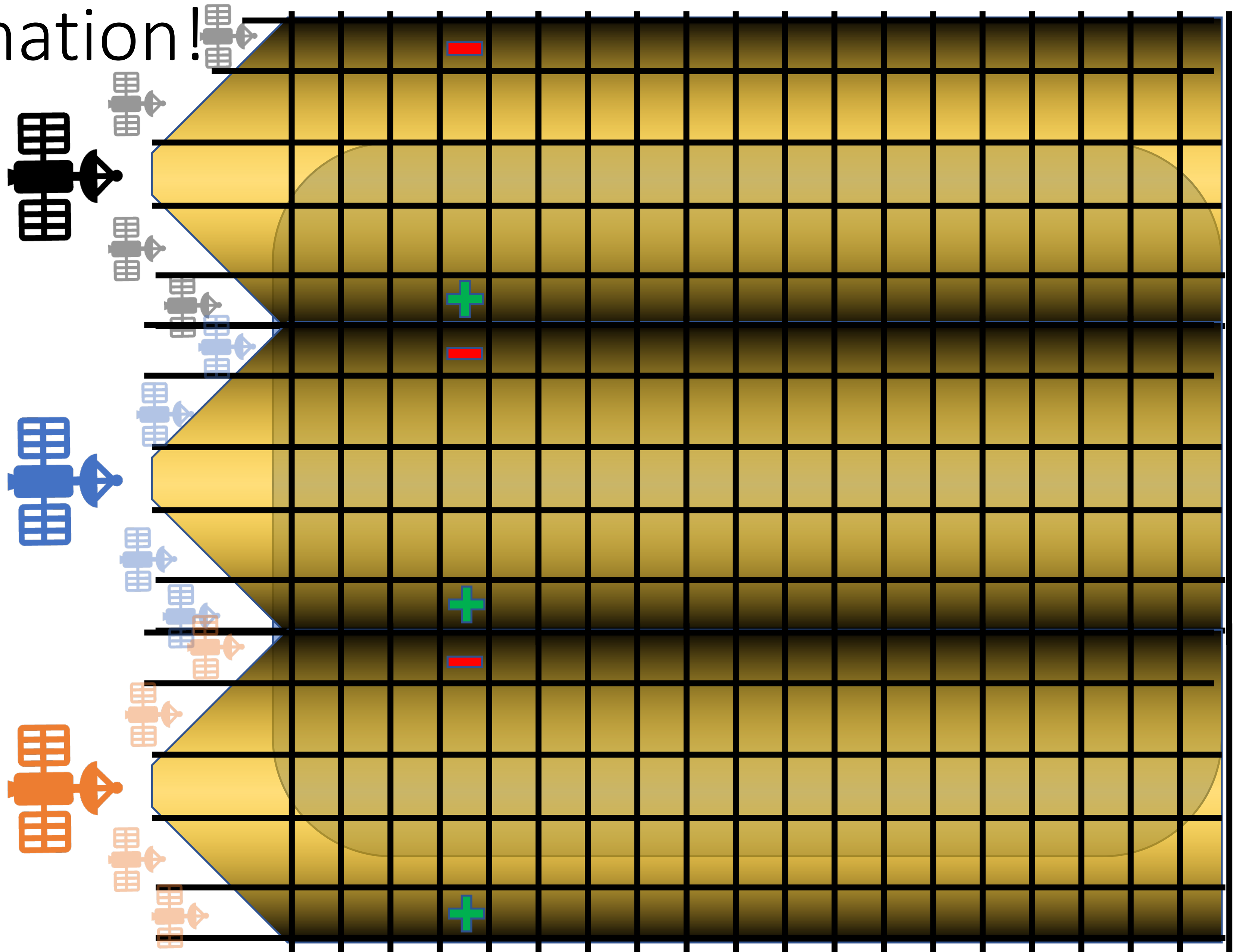


We can fix this using Doppler information!

But!! Now we're not using all our bursts.

And we get the brightest returns from the center (zero-Doppler), where the azimuth beam pattern is maximum, so some areas of the image will be brighter than others.



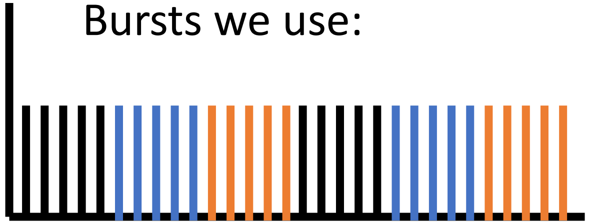


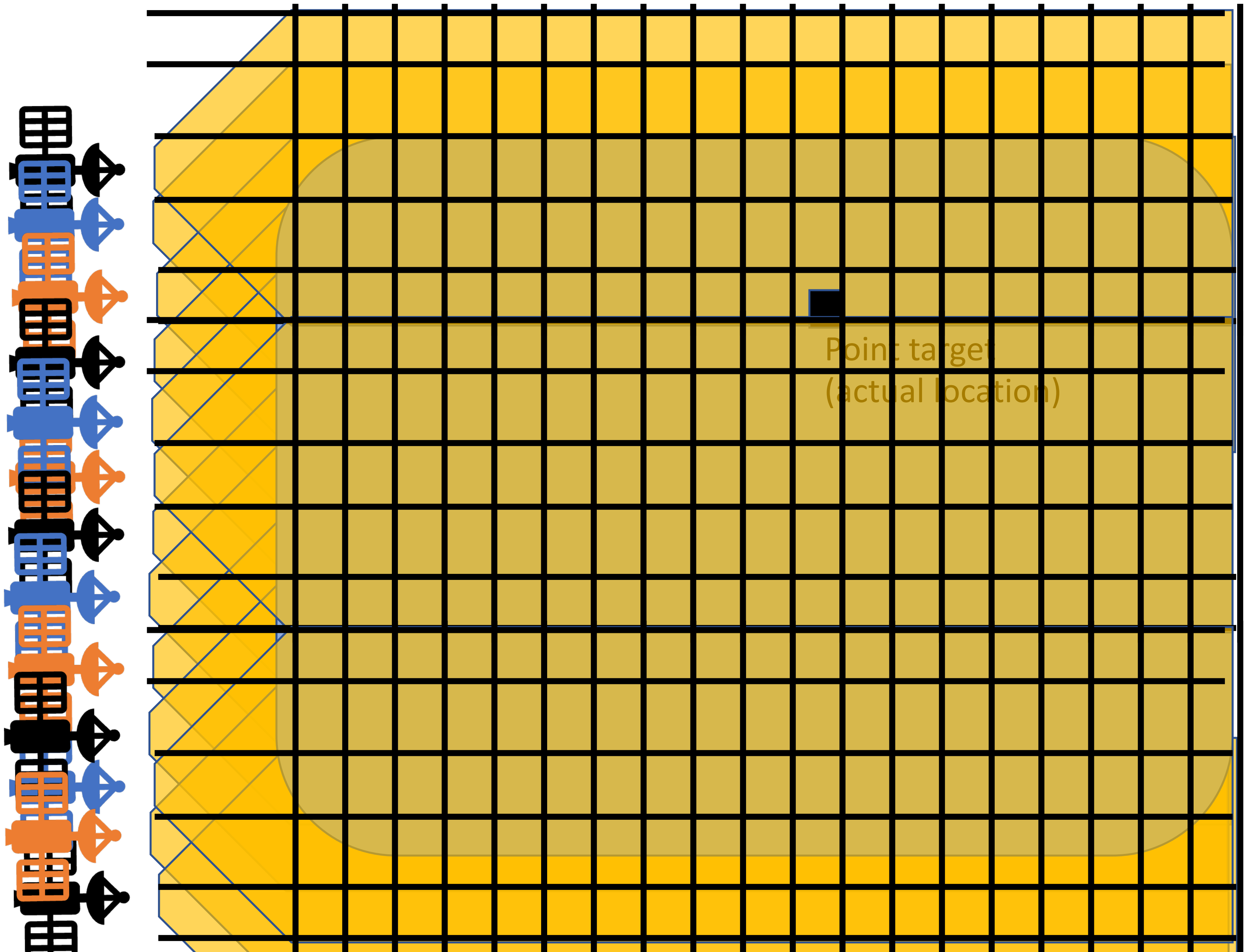
Now, let's use all pulses

We use a burst of e.g. 64 pulses at a time.

Within a pixel, we can add up results from multiple, overlapping bursts that image that pixel (e.g. have a response in that range bin, at that Doppler frequency).

Bursts we use:



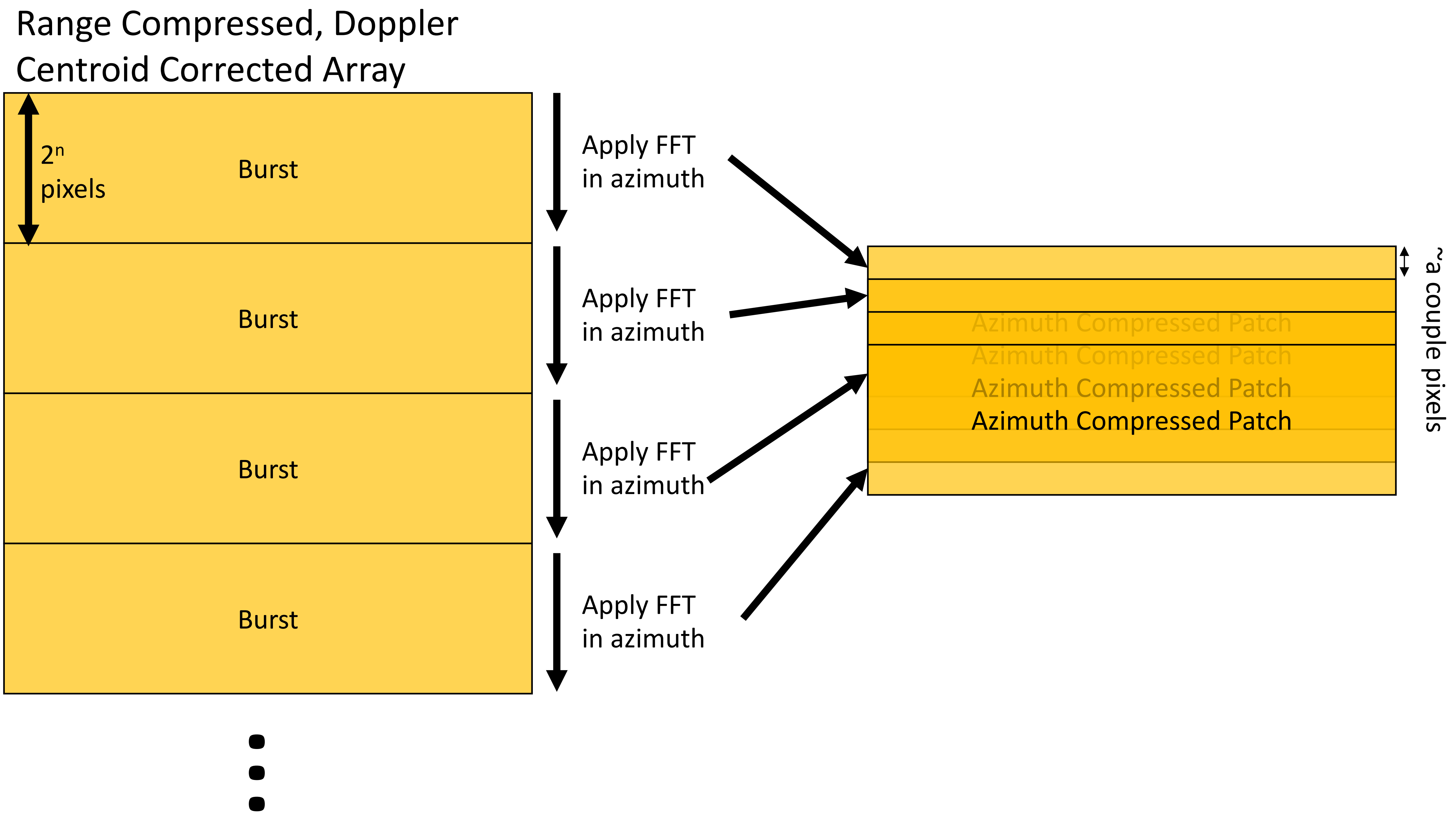


Some implementation concerns

- What is the resolution?
  - We can derive $\delta_{az} = \sqrt{\lambda r_0}$ (In our example this was 219 m).
  - This is a physical limitation of the radar with this processing style: we *could* oversample with this method, but it is pointless if we can't resolve more.
- How does time between pulse transmission come into this?
  - The distance between pulses is set by the PRF (pulse repetition frequency).
  - Pulse spacing $\Delta p = \frac{v}{PRF}\ (m)$ For example, pulses may be 5 m apart.
- How many pulses do we include in each "burst" subset?
  - # pulses per burst $n_p > \frac{\delta_{az}}{\Delta p}$. Round up to next power of 2 for FFT convenience.
  - Then each burst covers $n_p\ \Delta p$ m on the ground. e.g. $= (64\ pulses)(5\ m\ apart)$
- How do we know how much adjacent "burst" sets of pulses overlap? →

Some implementation concerns

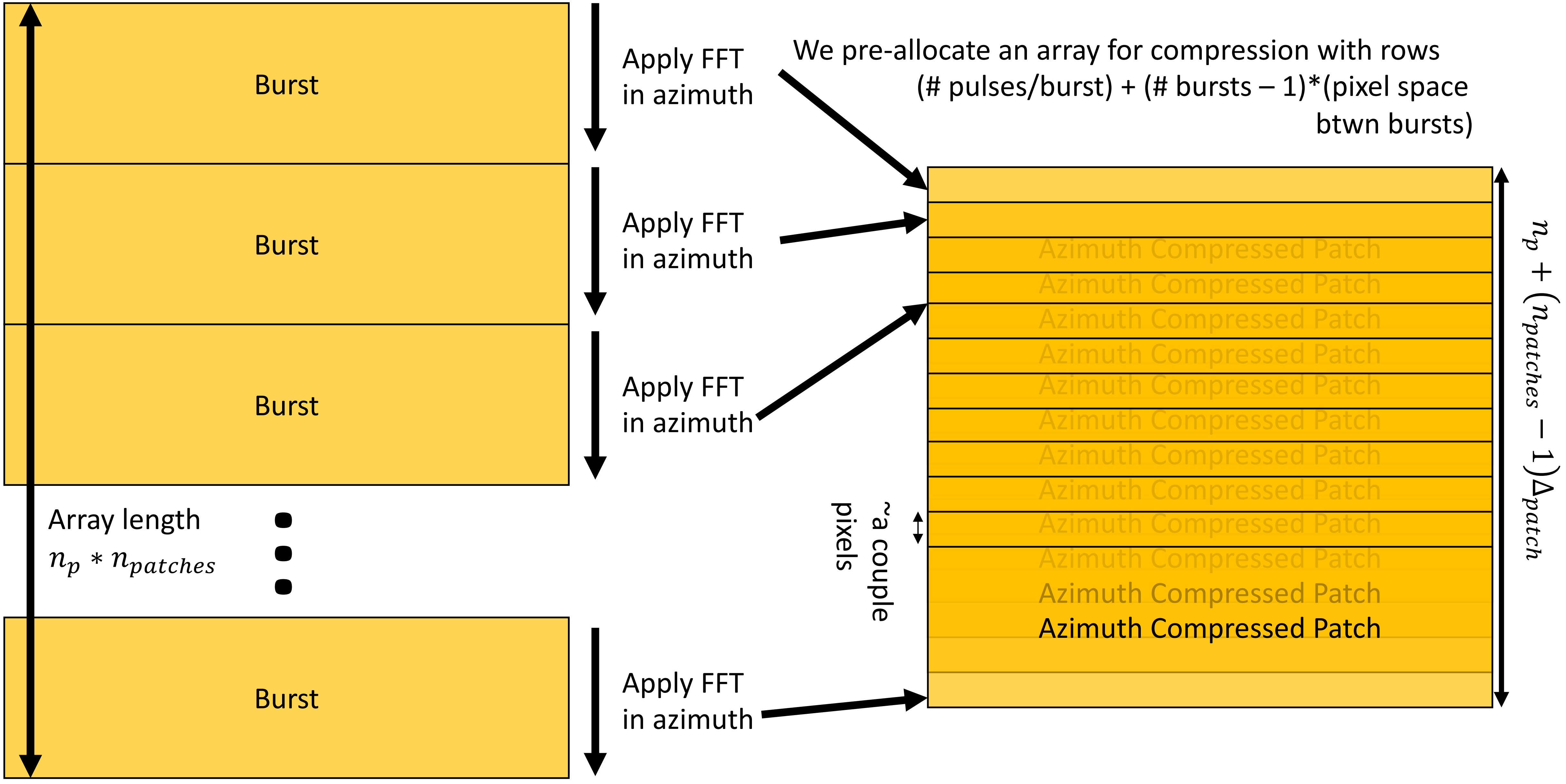
- How do we know how much adjacent “burst” sets of pulses overlap?
  - Patch-to-patch spacing.
  - $\Delta_{\text{patch}} = \frac{(\Delta p)(n_p)}{\delta x} = \frac{(pulse\ spacing\ in\ az)(num\ pulses)}{(output\ pixel\ spacing)} \cong a\ few\ pixels$
  - So, for each burst (e.g. of 64 pixels) we collect, we calculate the FFT in azimuth, and then place the output mostly overlapping the previous burst, but a few pixels down. Where it overlaps, add the multiple bursts together.
  - **We use completely unique, non-overlapping pulse trains for each burst.**
  - **Our results, however, do overlap.**
- Note frequency resolution (of FFT) $\delta f = \frac{PRF}{np}$ (PRF is BW, np is # pulses)
- Note output pixel spacing $\delta x = \frac{(\delta f)\lambda r_0}{2v}$ (e.g. 80 m)


Range Compressed, Doppler Centroid Corrected Array
2^n pixels
Burst
Burst
Burst
Burst
Apply FFT in azimuth
Apply FFT in azimuth
Apply FFT in azimuth
Apply FFT in azimuth
Azimuth Compressed Patch
Azimuth Compressed Patch
Azimuth Compressed Patch
Azimuth Compressed Patch
~a couple pixels

Range Compressed, Doppler Centroid Corrected Array

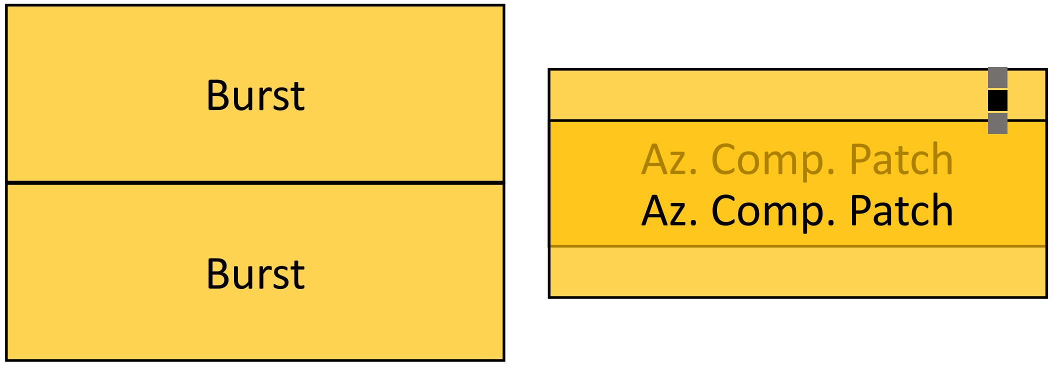
Originally, our array has a number of rows equal to the total number of pulses: (# pulses/burst)*(# bursts)

We pre-allocate an array for compression with rows (# pulses/burst) + (# bursts – 1)*(pixel space btwn bursts)



How good is our focusing?

- With unfocused SAR, our burst might include, e.g., 32 or 64 pulses.
  - Our minimum number of pulses per burst, $n_p > \frac{\delta_{az}}{\Delta p}$, is set by the azimuth resolution of the system and the physical spacing between pulses. We want at least as many pulses as will fill one resolution cell ("resel") in a burst.
  - Our maximum number of pulses per burst is constrained because the platform moves while we're acquiring each pulse. If the system moves more than a pixel during a burst, the targets get smeared across the number of pixels traversed.
  - Real-aperture radar smears across whole beam, but unfocused algorithm smears across distance travelled during each small patch



Point target smeared across a couple pixels in output image as platform moves during burst

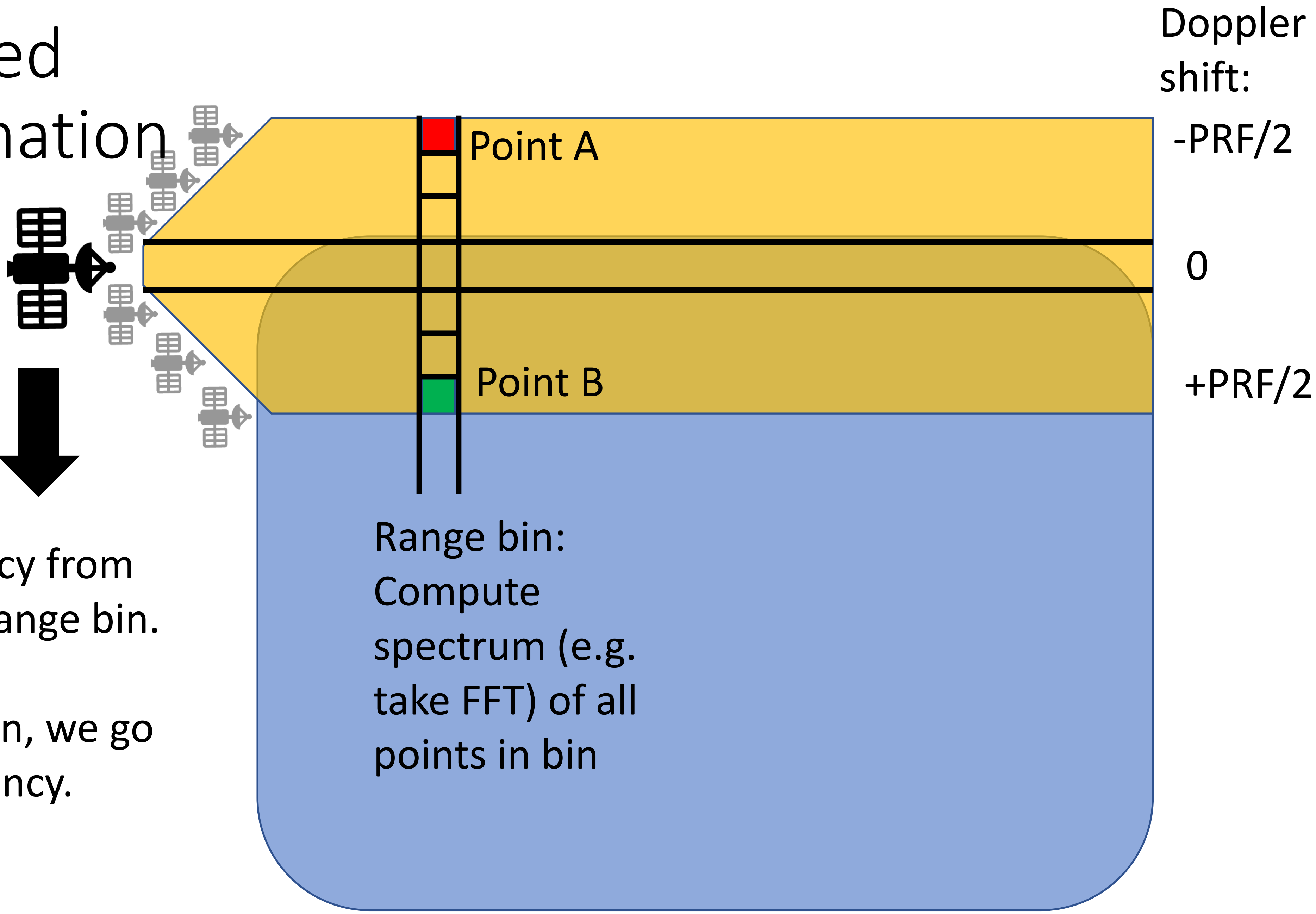
How do we fix this few-pixel blurring? Time for **focused SAR**!

We implement this by knowing more precise information about the phase of a point as the radar flies by.

We don’t have to stop with an FFT in azimuth – we can create a **matched filter** to the azimuth spectrum!

Recall our sorted Doppler information

We found frequency shift within a pulse series by taking a Fourier transform



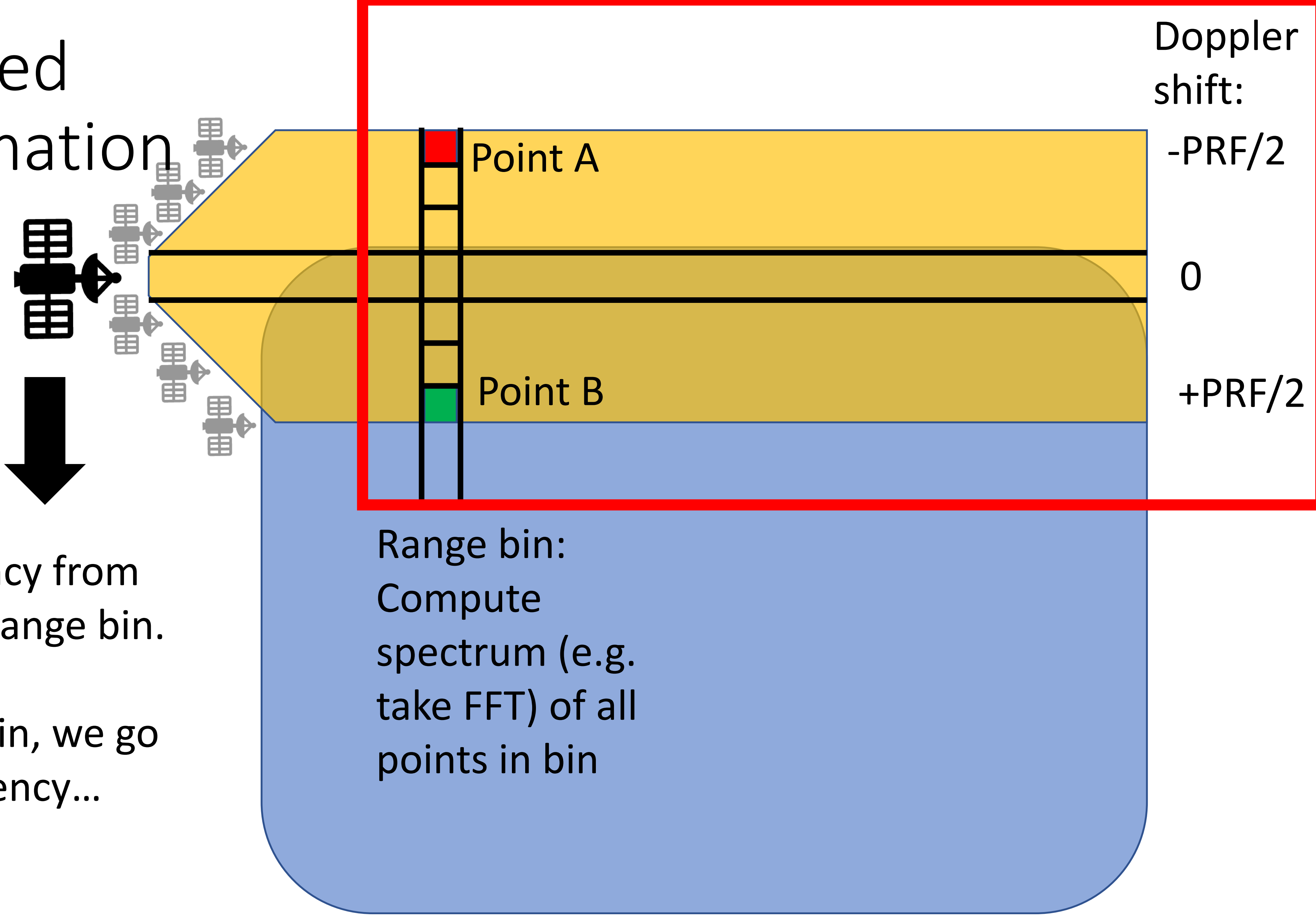
This sorted the frequency from low to high across the range bin.

So as we traverse the bin, we go from low to high frequency.

Let’s zoom in…

Recall our sorted Doppler information

We found frequency shift within a pulse series by taking a Fourier transform



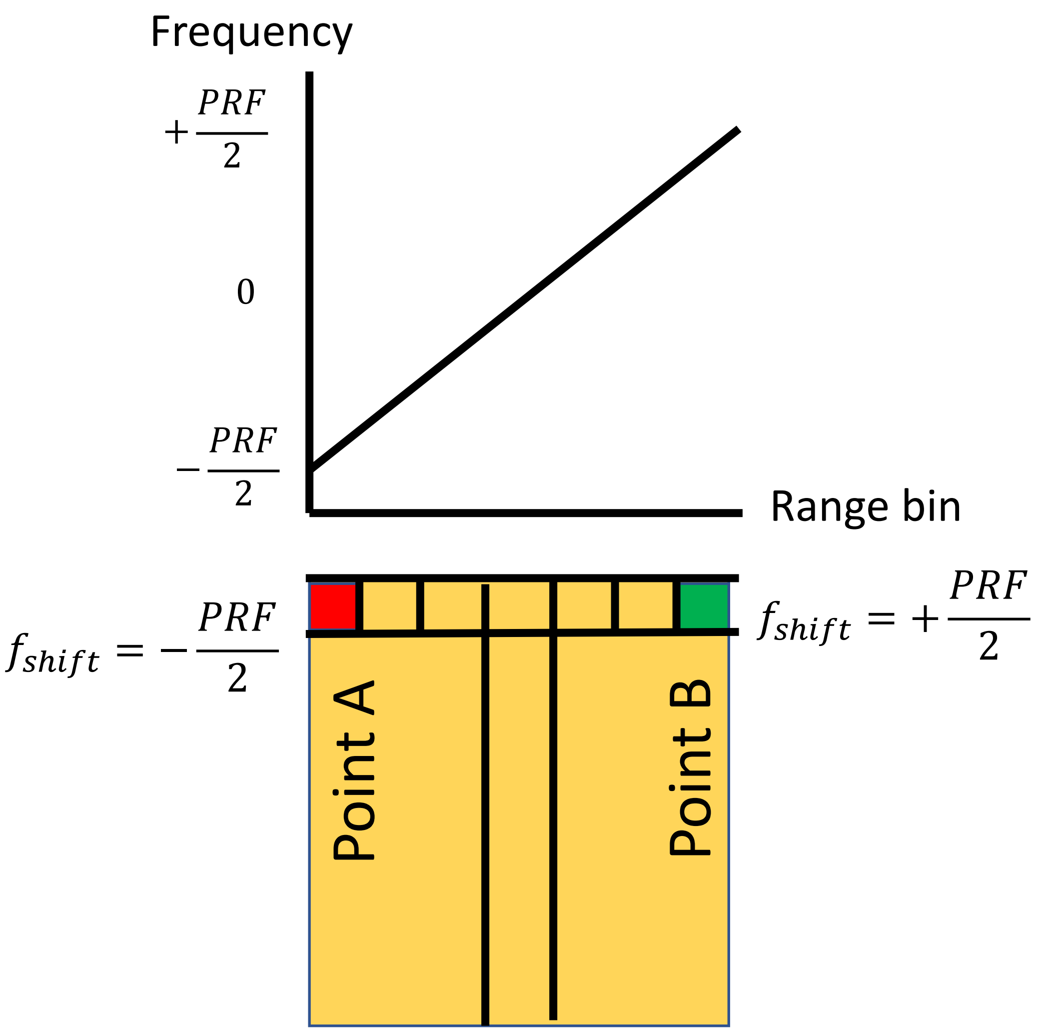
This sorted the frequency from low to high across the range bin.

So as we traverse the bin, we go from low to high frequency…

Let's zoom in

Our azimuth spectrum looks like a chirp across range bins!

- Across the azimuth bins, we see a linear change in frequency
- What shape is a linear change in frequency?
  - A chirp!
- We used the chirp shape of the range data to range compress with a matched filter
- We can do the same thing in azimuth!

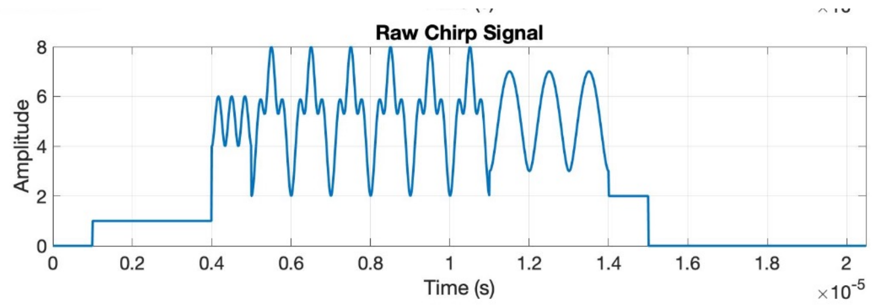


Can we make a matched filter in azimuth too? Yes, if we know the parameters!

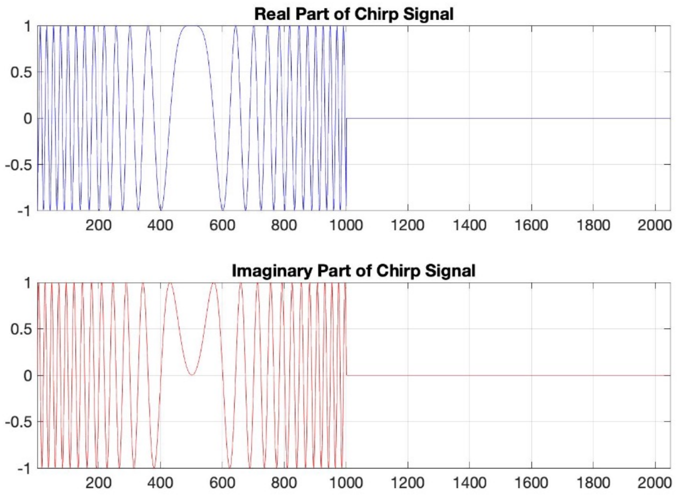
- Azimuth response of radar echo is quadratic in time – like a chirp:
  - $\phi(t) \cong -\frac{2\pi}{\lambda}\frac{v^2}{r_0}\boldsymbol{t^2}$
- We call chirp slope in azimuth **chirp rate** $\left(f_R = \frac{2v^2}{\lambda r_0}\right)$
- We can find bandwidth by multiplying chirp rate by pulse length (here, pulse length = time that an area is illuminated, $\tau = \frac{r_0\lambda}{v\ell}$):
  - $BW = f_R\tau = \frac{2v^2}{\lambda r_0}\,\frac{r_0\lambda}{v\ell} = -\frac{2v}{\ell}$. So, our resolution in time $\delta t = \frac{1}{BW} = -\frac{\ell}{2v}$
  - And spatial resolution $\delta_{az} = v\,\delta t = \frac{\ell}{2}$!

By making a matched filter, we are no longer limited to taking FFT’s covering the same area

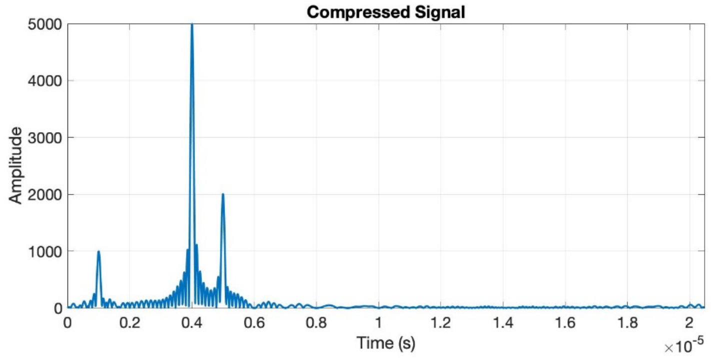
- We can theoretically take an FFT in azimuth over the WHOLE image and filter it, and get a very finely focused output!
- In the HW, we have you focusing patches of size 2048 (just for the sake of processing speed), but this can be as big as you want



*



=



The idea of focused SAR is that we can use the parameters of the system (range, velocity, and wavelength) to focus an image to the finest resolution possible

This matched filter is powerful!

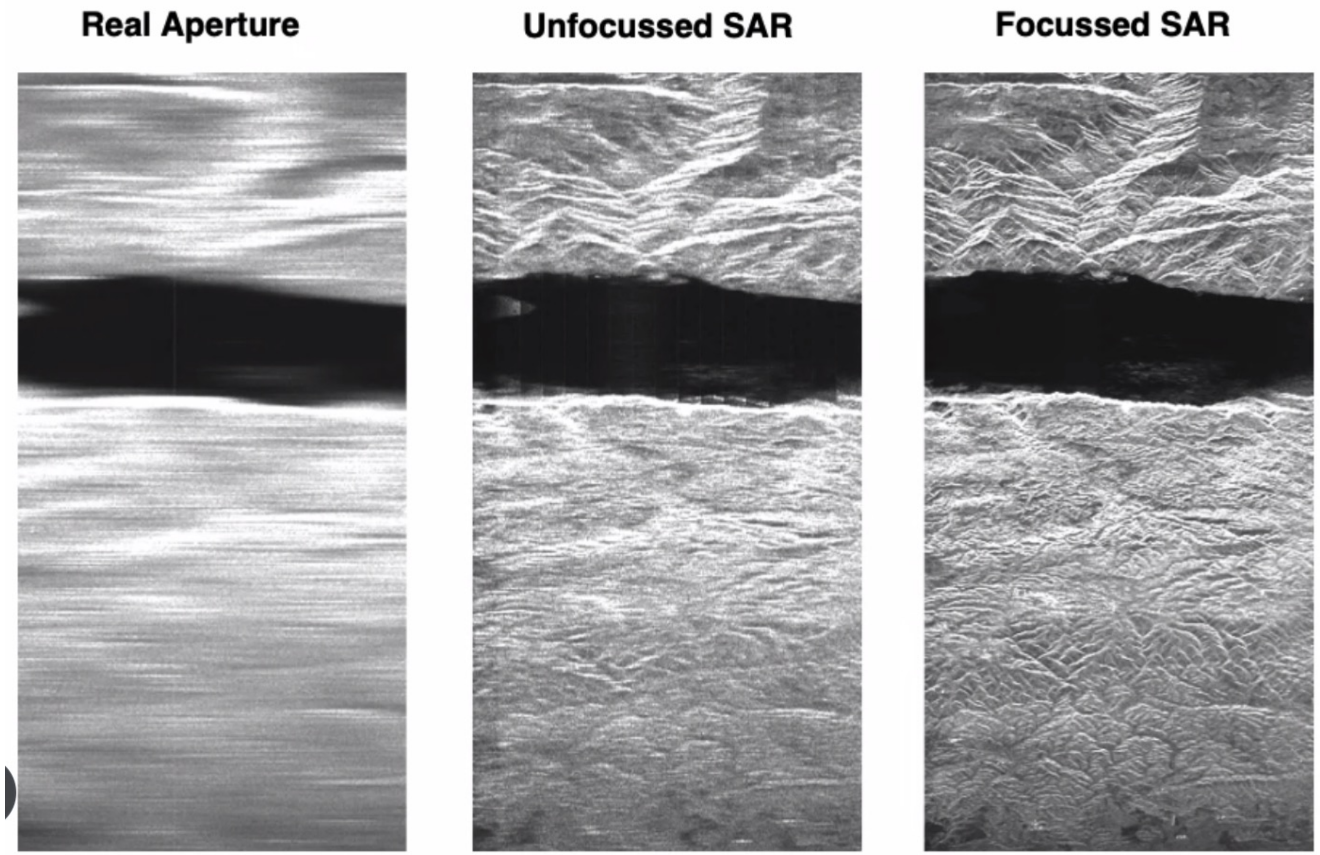
SAR Processing Algorithm	Azimuth Resolution δ_{az}	Example value
Real Aperture Radar	$\frac{r\lambda}{\ell}$	$100\ m$
Unfocused SAR	$\sqrt{\lambda r_0}$	$10\ m$
Focused SAR	$\frac{\ell}{2}$	$0.5\ m$

Example:

$r = 2\ km$

$\lambda = 5\ cm$

$l = 1\ m$


Real Aperture
Unfocussed SAR
Focussed SAR