How CRACO works

An introduction for astronomers and commission…ers

Keith Bannister 25 Aug 2021 & 31 Nov 2021

Aim

- Antenna-coherent FRB search - 5x improvement in detection sensitivity
- Similar time/frequency resolution to existing ICS system: ~1ms/1MHz
- 0.5-2 FRBs / day each with ~arcsecond localisation
- Re-use existing voltage dump infrastructure
- Just build an awesome detection machine

History

GPU vs FPGA

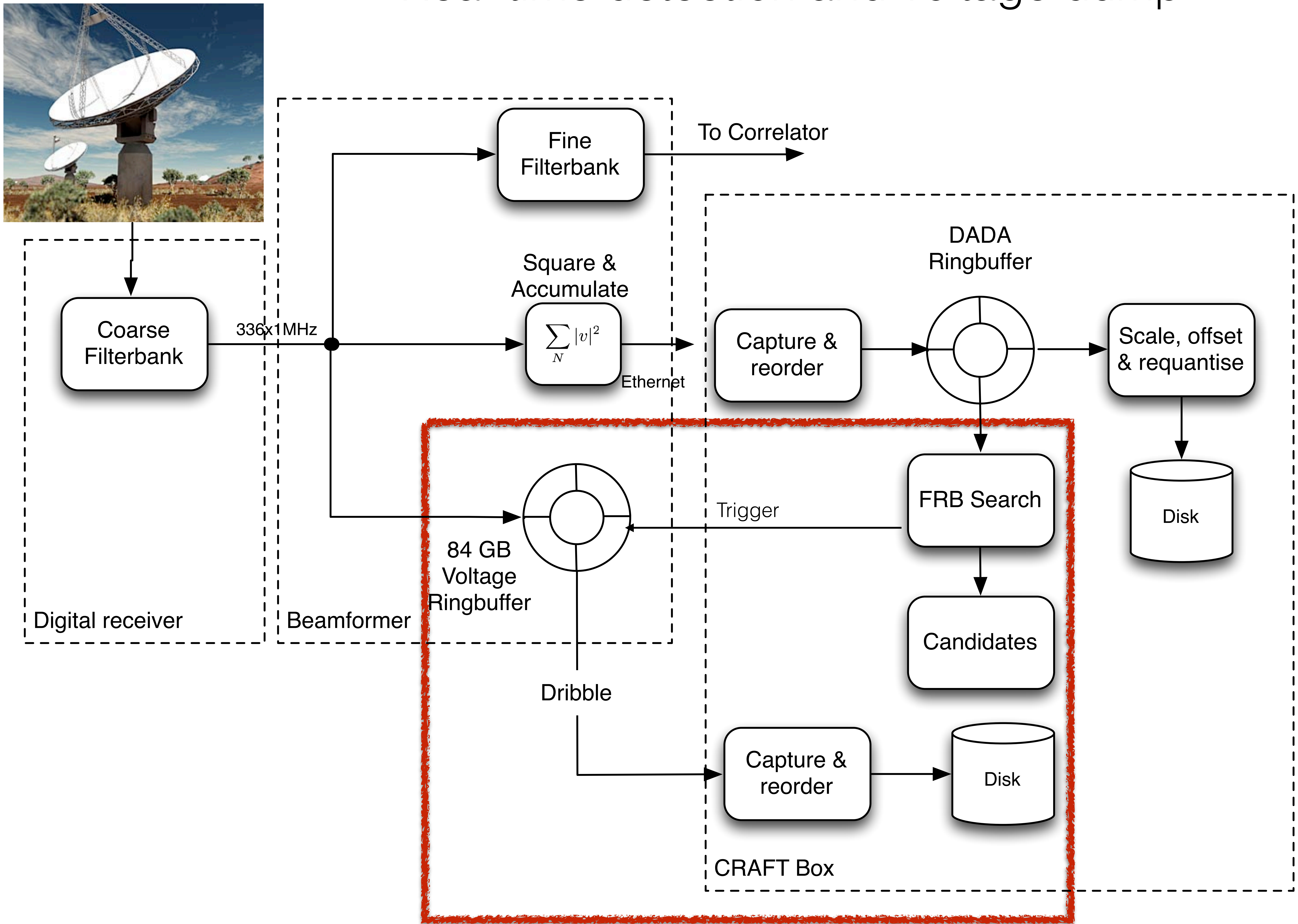
A stressful decision

- Talked about CRACO at ASA 2018
- Preliminary study / review thingy July 2019
- Started benchmarking technologies Dec 2019
- Committed to FPGAs in COVID madness March 2020. J-P: “High risk, high return”
- Pilot cluster order placed April 2020
- Pilot cluster commissioned ~April 2021
- Working processing pipeline - Aug 2021
- Commissioning in lab: TBC
- First light TBC

Scientific Uses

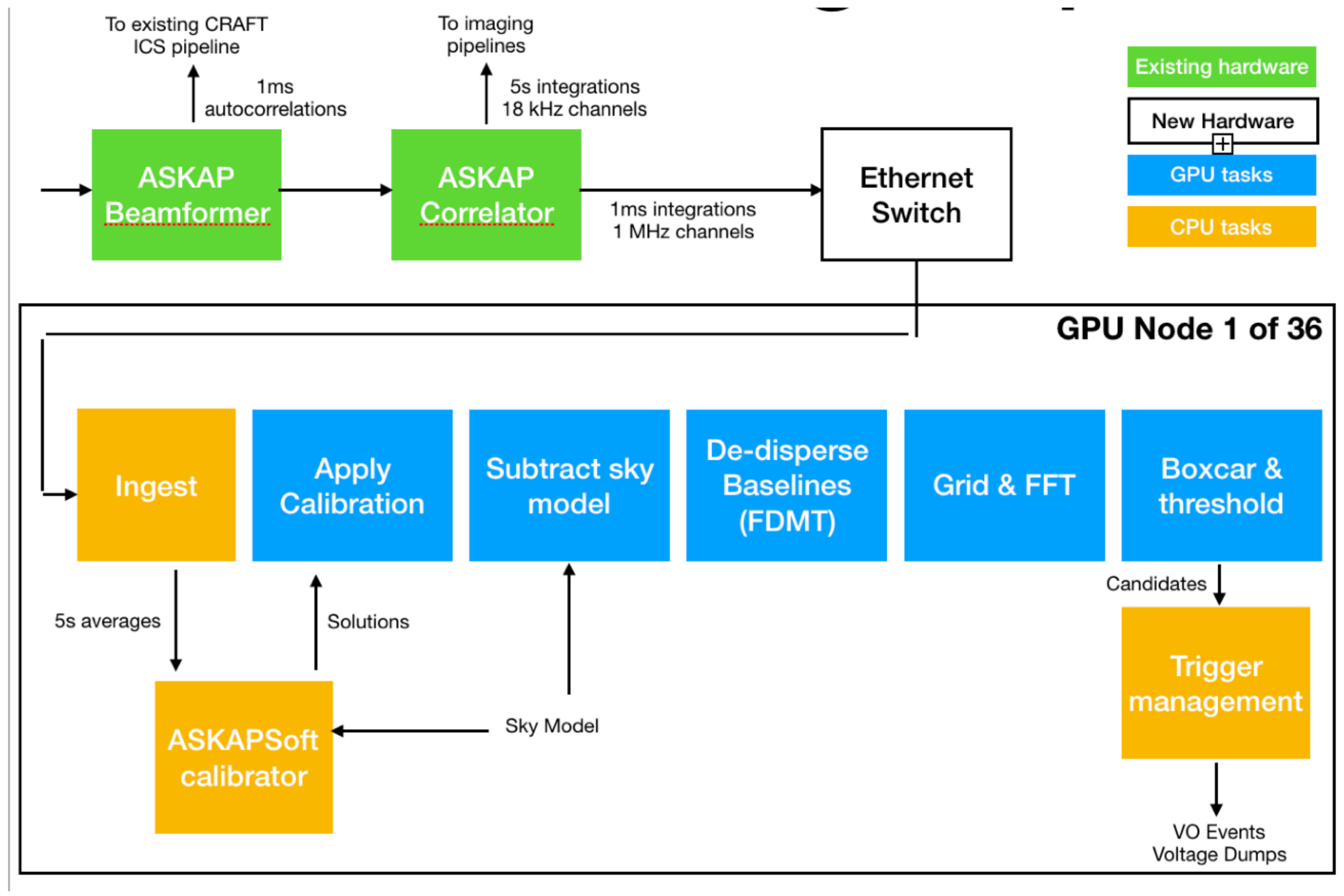
- Large numbers of FRBs: Large statistical sample, cosmology, weird guys
- Higher redshift FRBs: - strongest constraints on cosmology and the early universe
- Interplanetary scintillation of AGN- space weather
- Dual-use as VLBI recording (online channel stitching)
- Pulsar?
- SETI?

Real-time detection and voltage dump

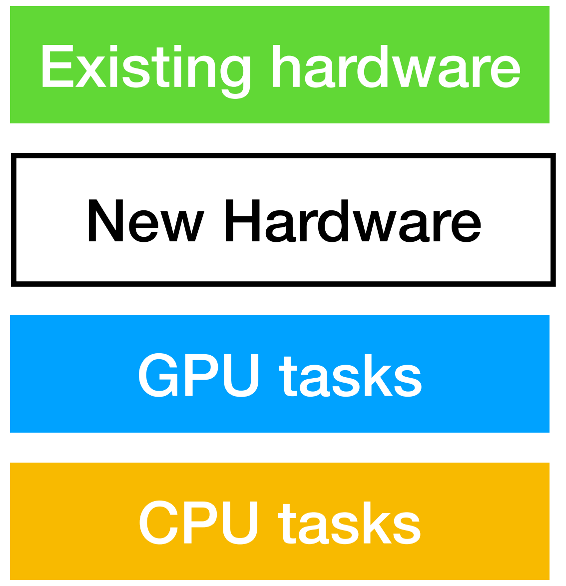


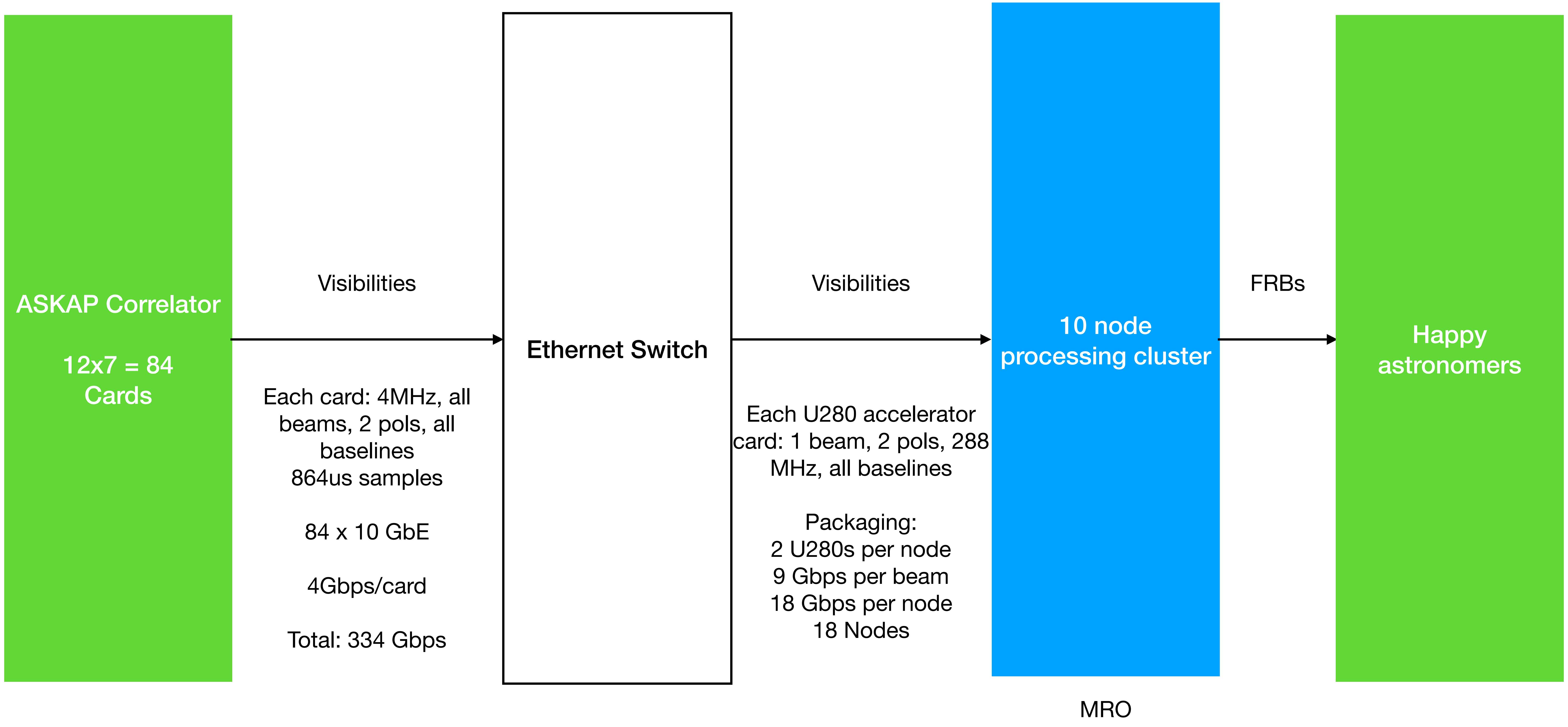
CRACO

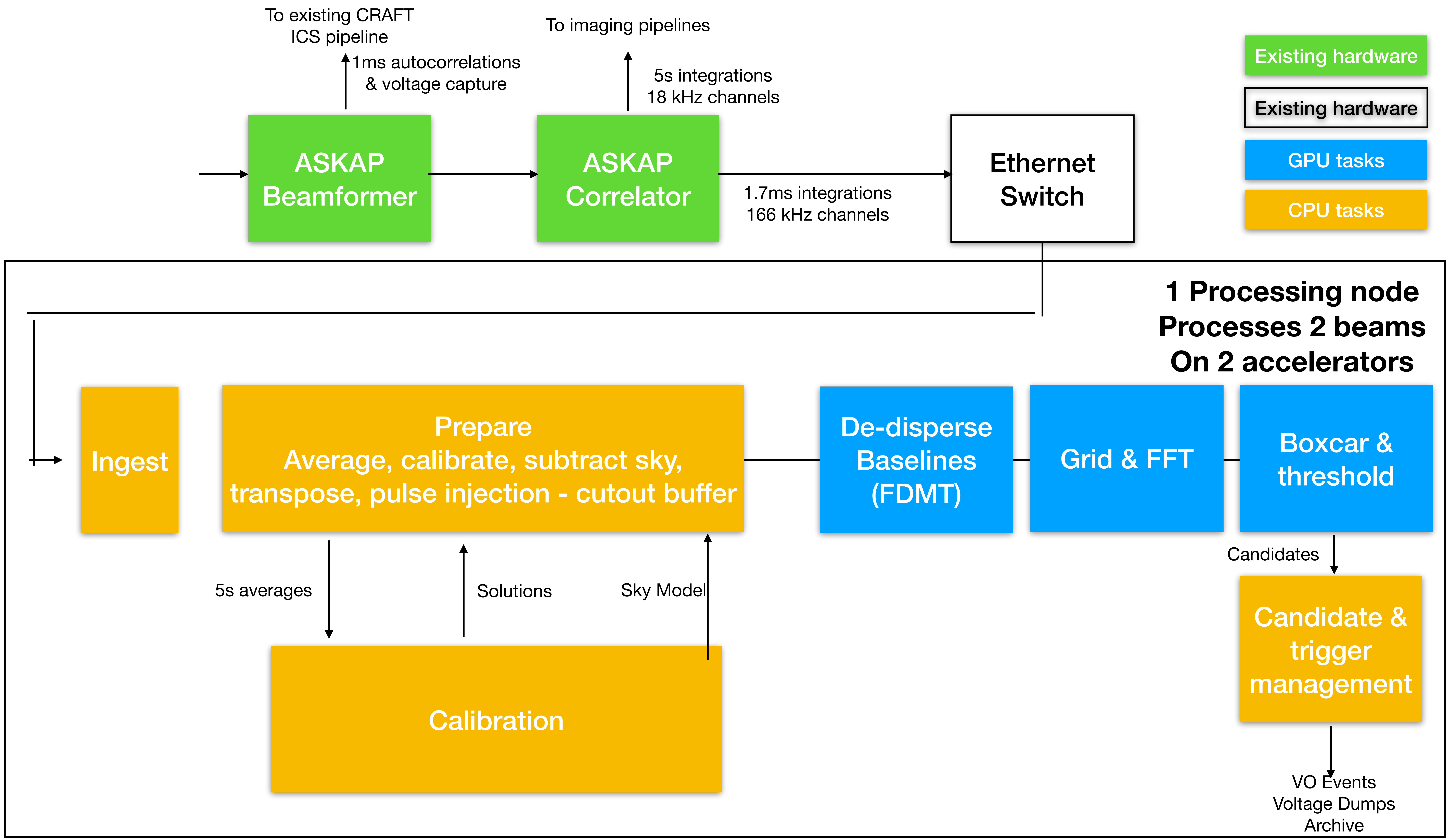
- Antenna-coherent mode:
  - ~$1M cluster
  - 5x improvement in sensitivity,
  - Full FoV
  - Form and watch 23 Terapixels per second
  - = 0.8 Million netflix watchers
  - Increase detection rate: 0.5-2 FRBs/day



Pilot cluster at MRO - installed

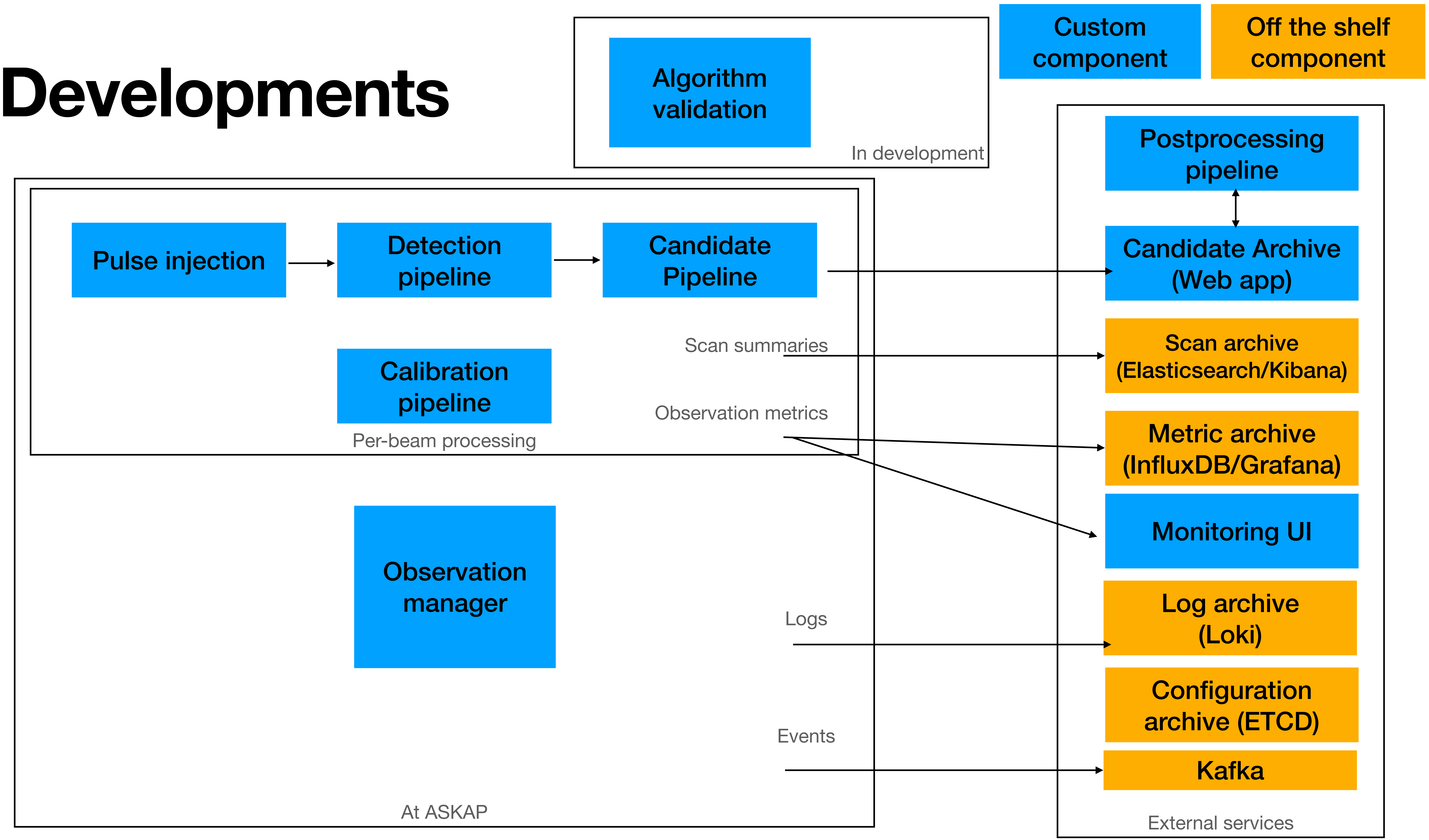






To existing CRAFT
ICS pipeline
1ms autocorrelations
& voltage capture
To imaging pipelines
5s integrations
18 kHz channels
Existing hardware
Existing hardware
GPU tasks
CPU tasks
ASKAP
Beamformer
ASKAP
Correlator
1.7ms integrations
166 kHz channels
Ethernet
Switch
1 Processing node
Processes 2 beams
On 2 accelerators
Ingest
Prepare
Average, calibrate, subtract sky,
transpose, pulse injection - cutout buffer
De-disperse
Baselines
(FDMT)
Grid & FFT
Boxcar &
threshold
Candidates
Candidate &
trigger
management
5s averages
Solutions
Sky Model
Calibration
VO Events
Voltage Dumps
Archive

Developments

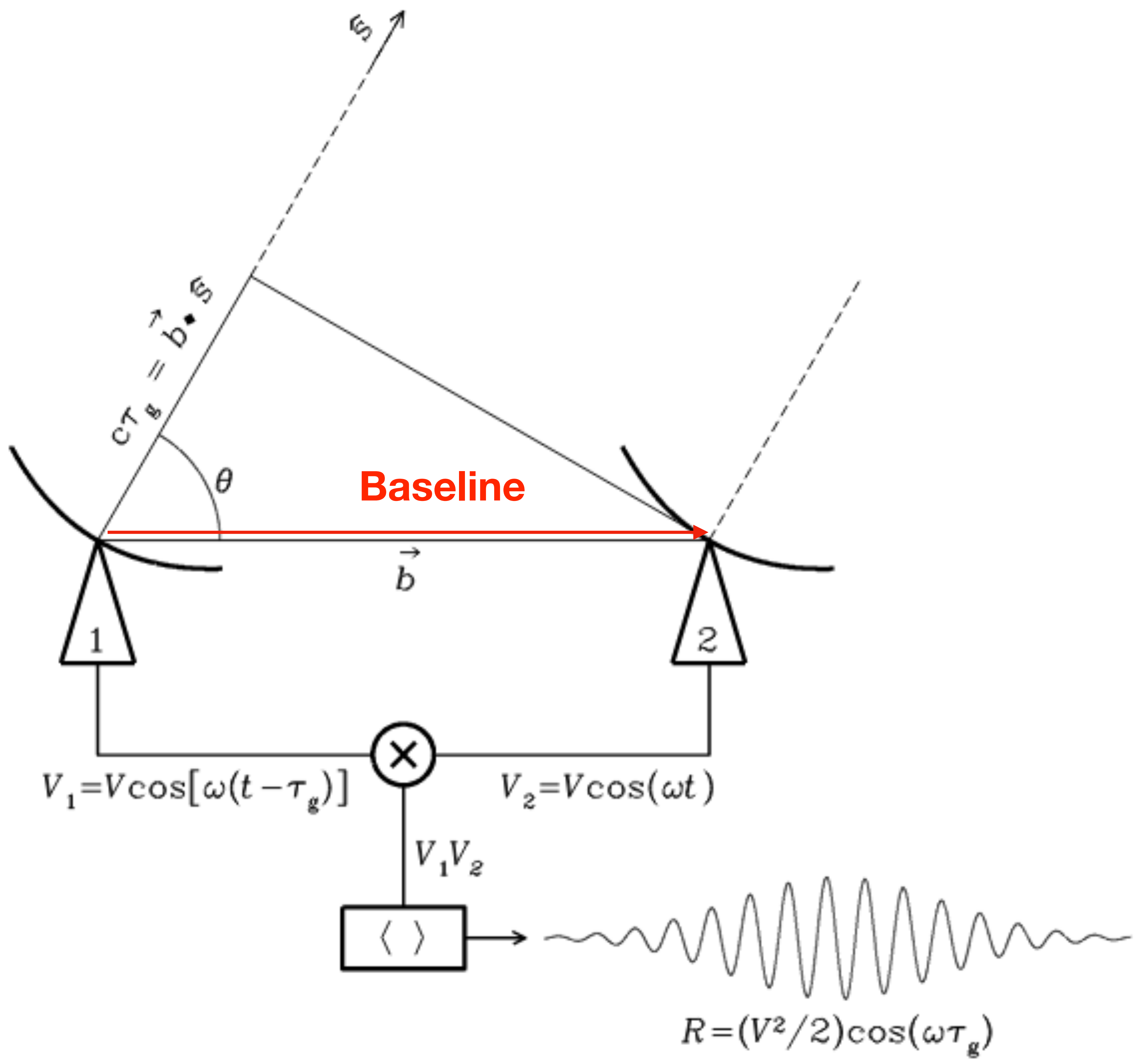


Variables
For reference

Symbol	Description	Typical value
NT	Number of samples per block	256
NDM	Number of DM trials to process	512
MAX_DM	Largest DM to process (in samples)	1024
NC	Total number of channels	288
NBL	Total number of baselines	435
NCIN	Number of channels on FDMT input	32
NDOUT	Number of DMs on output of FDMT	186
Tsamp	Integration sample time	1.7ms
NUV	Number of UV cells to process	4000-8192 (max)
NUVWIDE	Number of UV cells to process at a time	8
NUREST	Number of blocks of uv cells to process	NUV/NUVWIDE

Background: Interferometry

- Thompson, Moran and Swenson (free PDF) is the standard reference
- See also the NRAO radio school notes
- Interferometers form ‘visibilities’ = the cross correlation between pairs of antennas
- The vector between a pair of antennas is called a “baseline” - which is given the coordinates b = [u, v, w] in a Euclidian coordinate system - measured in wavelengths
- W is not important for us and can be assumed to be 0 here.
- For N antennas, there are N(N-1)/2 baselines.



[u,v,w] coordinate system

Fig. 3.2 Geometric relationship between a source under observation $I(l, m)$ and an interferometer or one antenna pair of an array. The antenna baseline vector, measured in wavelengths, has length D_λ and components (u, v, w).

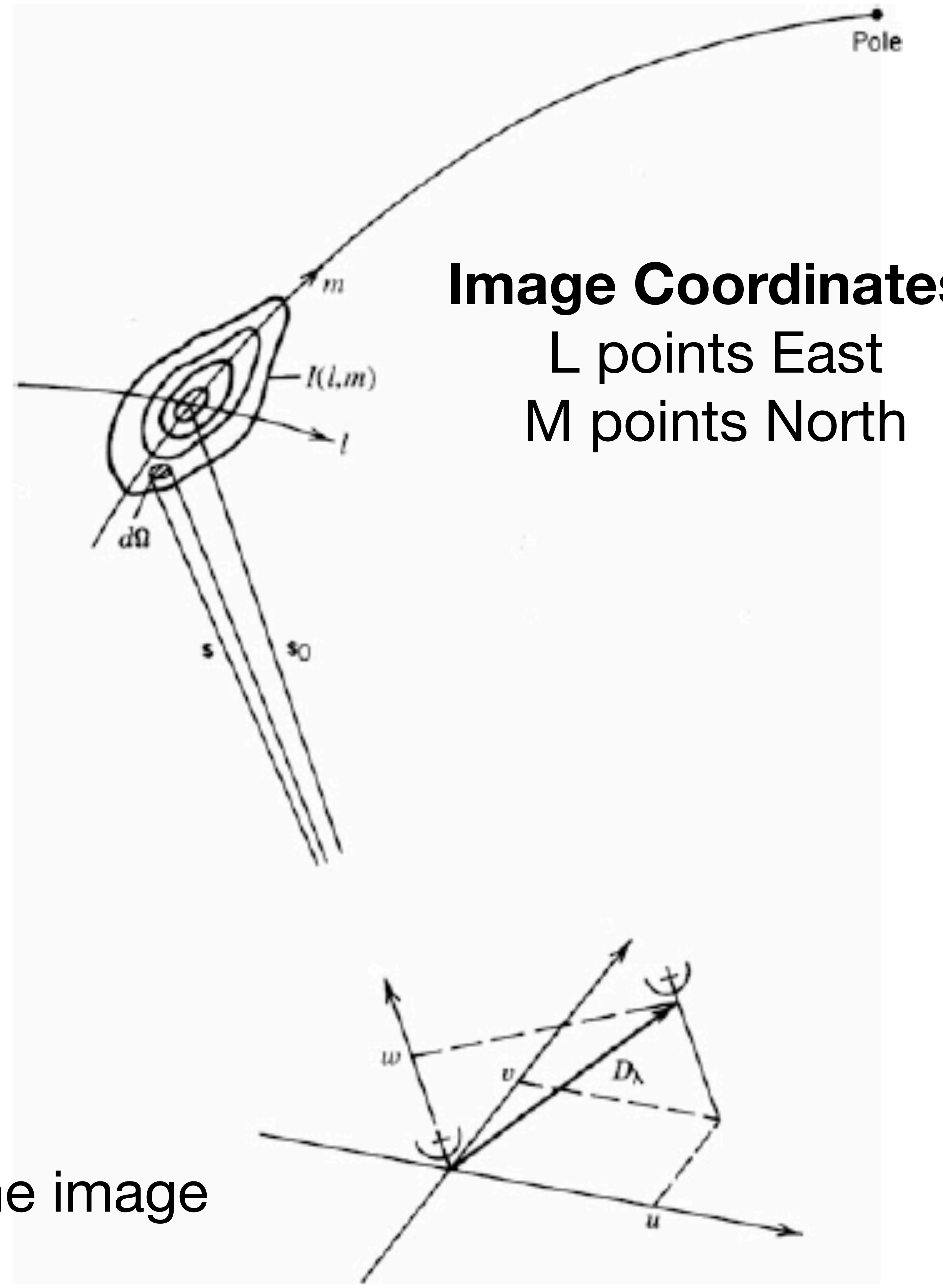
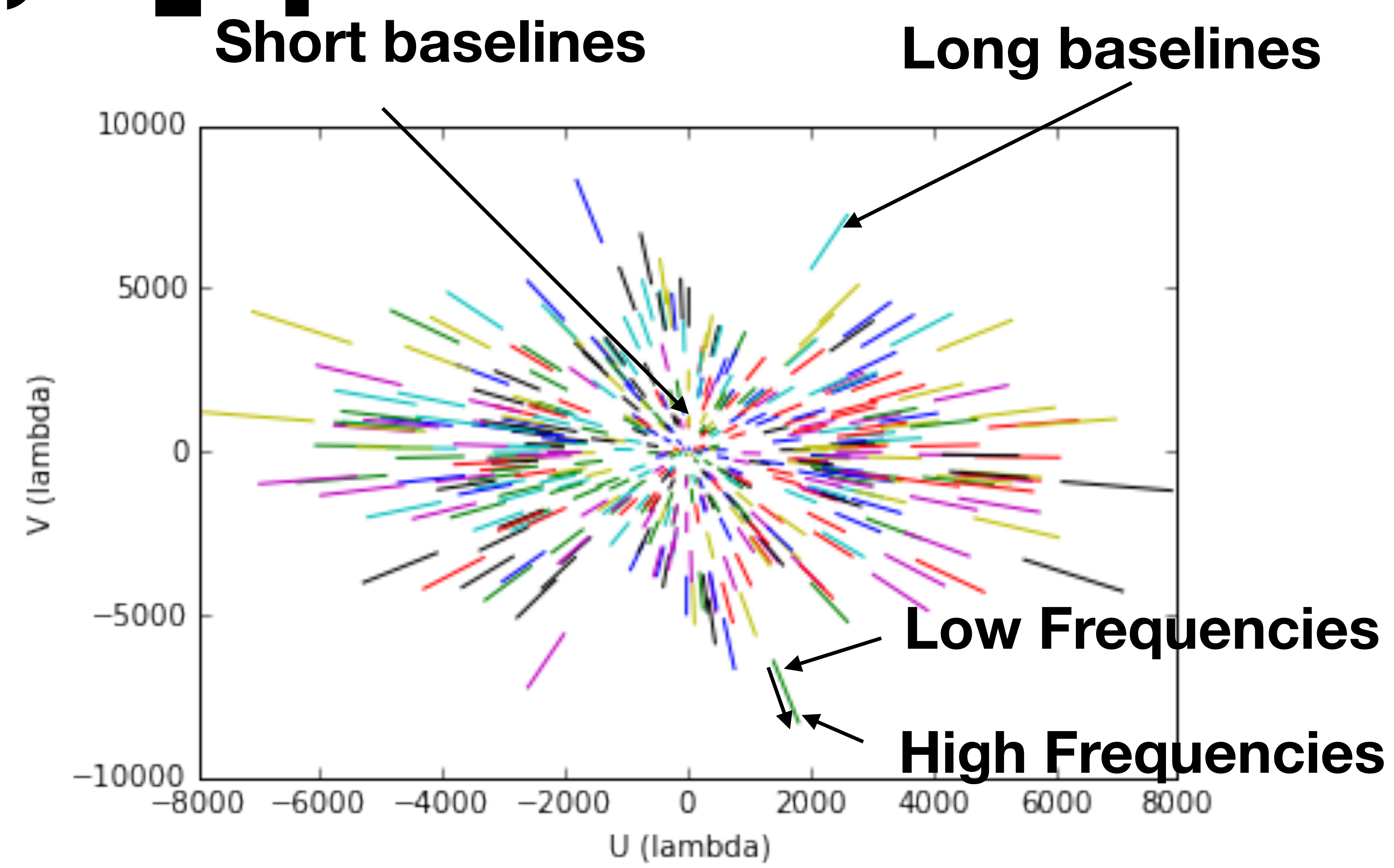


Image Coordinates:
L points East
M points North

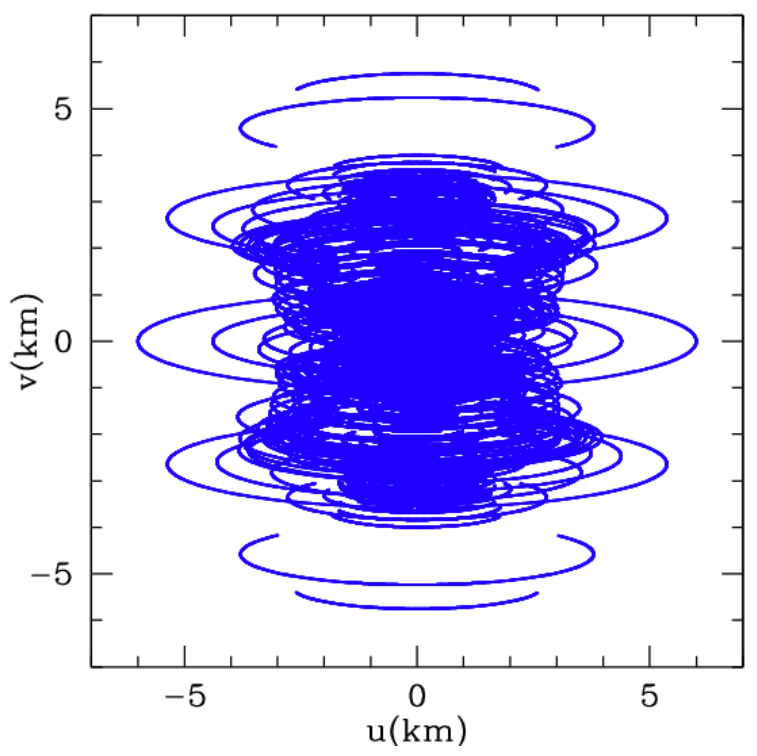
Baseline Coordinates:
W points towards the centre of the image
U points East
V points North

The [u,v] plane

- At any given moment, a baseline (for a single frequency) will occupy a single point in [u,v] (ignore w)
- Baselines with small distance between then will appear near the centre of the UV plane.
- Baselines with a large distance will be further away.
- A single baseline with multiple frequency channels will appear as a radial line. The length of the line increases with baseline length.
- Baselines will rotate around the centre of the UV plane as the Earth rotates
- For a short integration the UV plane is often very sparse.



Typical UV plane for 22 ASKAP antennas
For 1 millisecond with all channels



Typical UV plane for 36 ASKAP dishes for a single frequency over 10 hrs

Imaging

- The correlation of signals on a baseline gives the “visibility” - V(u, v) - a complex number per baseline and per channel.

- UV plane is complex valued and discretely sampled.

- The visibility is the Fourier transform of the sky brightness I(l, m) at a particular location [u,v] - I.e. in the uv plane.

- So to find the sky brightness vs direction - an image I(l,m) we- put all visibilities in the right place on the UV plane, and take the inverse FFT!

- The a image of the sky is real-valued (brightness) - so the UV plane is Hermetian (i.e. the correlation between Ant1 and Ant2 is the complex conjugate of the correlation between Ant 2 and Ant 1)

$$\mathcal{V}(u,v,w) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A_N(l,m) I(l,m) \times \exp\left\{-j2\pi\left[ul + vm + w\left(\sqrt{1-l^2-m^2}-1\right)\right]\right\} \frac{dl\,dm}{\sqrt{1-l^2-m^2}} . \tag{3.7}$$

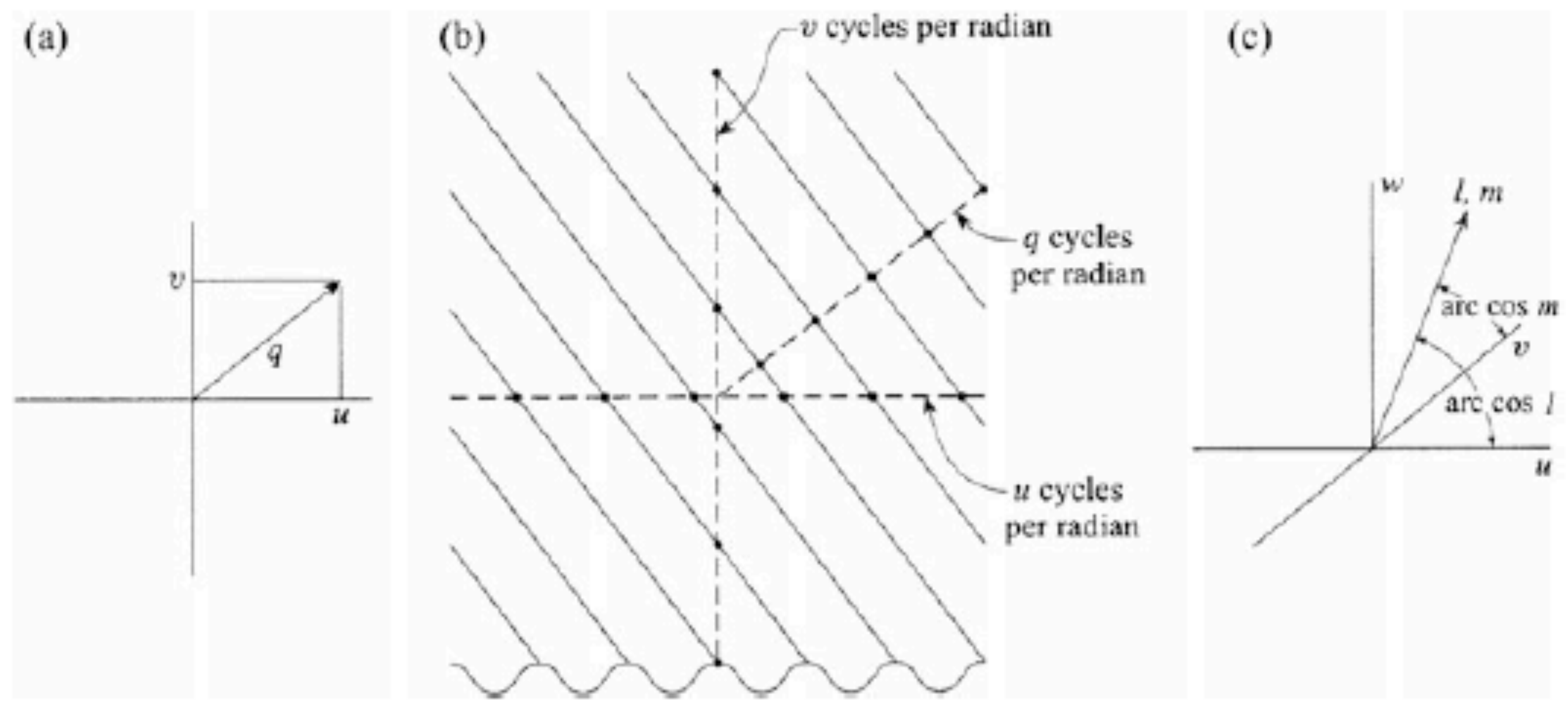
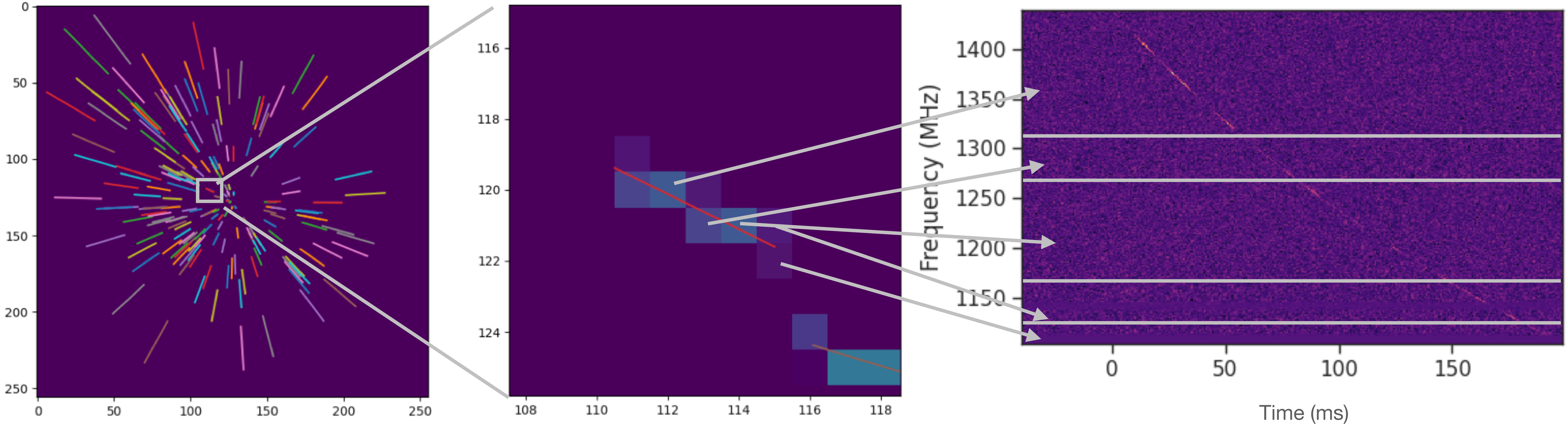


Fig. 2.7 (a) The (u, v) plane in which the arrow point indicates the spatial frequency, q cycles per radian, of one Fourier component of an image of the intensity of a radio source. The components u and v of the spatial frequency are measured along axes in the east–west and north–south directions, respectively. (b) The (l, m) plane in which a single component of spatial frequency in the intensity domain has the form of sinusoidal corrugations on the sky. The figure shows corrugations that represent one such component. The diagonal lines indicate the ridges of maximum intensity. The dots indicate the positions of these maxima along lines in three directions. In a direction normal to the ridges, the frequency of the oscillations is q cycles per radian, and in directions parallel to the u and v axes, it is u and v cycles per radian, respectively. (c) The u and v coordinates define a plane, and the w coordinate is perpendicular to it. The coordinates (l, m) are used to specify a direction on the sky in two dimensions. l and m are defined as the cosines of the angles made with the u and v axes, respectively.

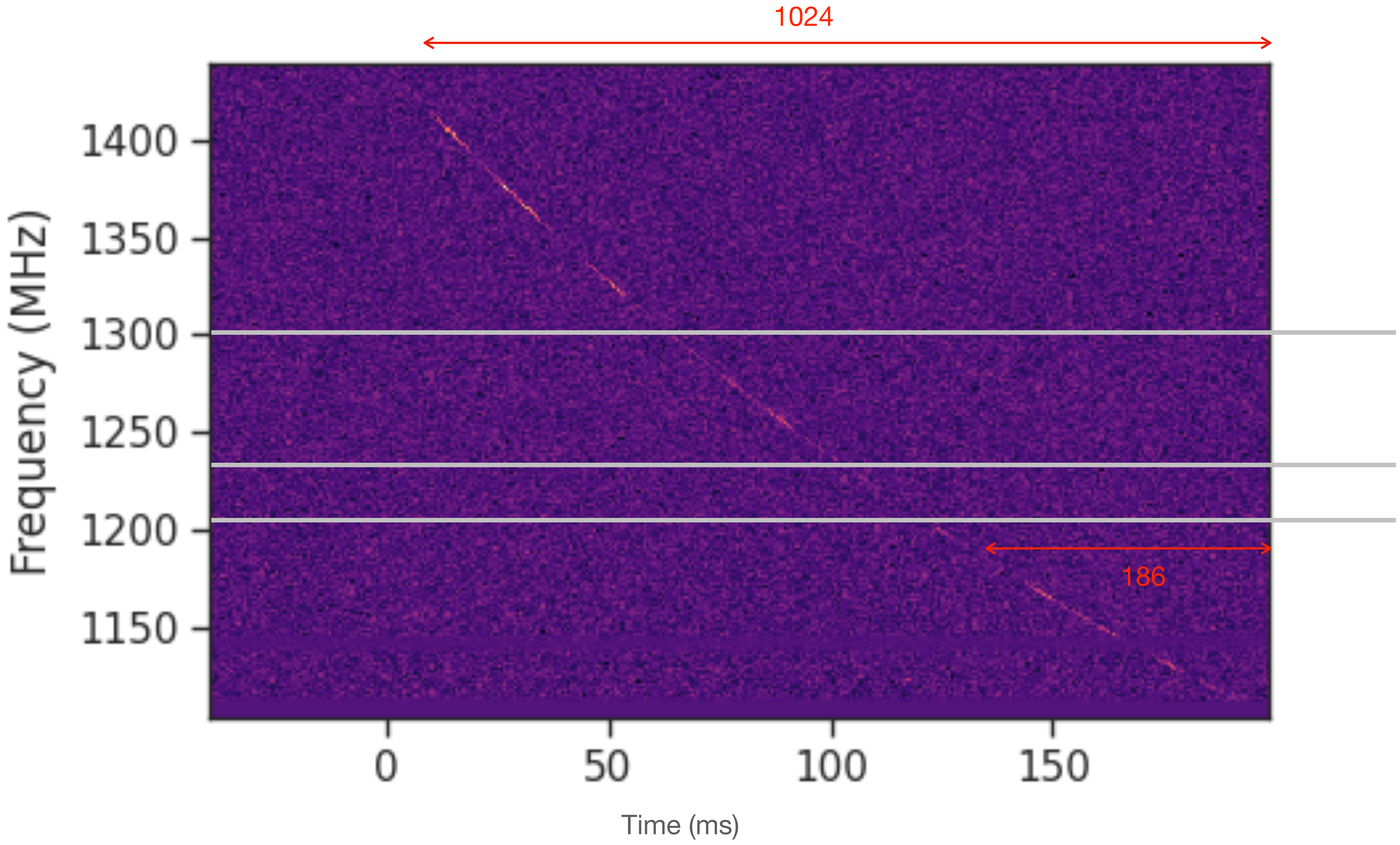
Different channels end up in different UV cells

We have to dedisperse them differently



Different channels end up in different UV cells

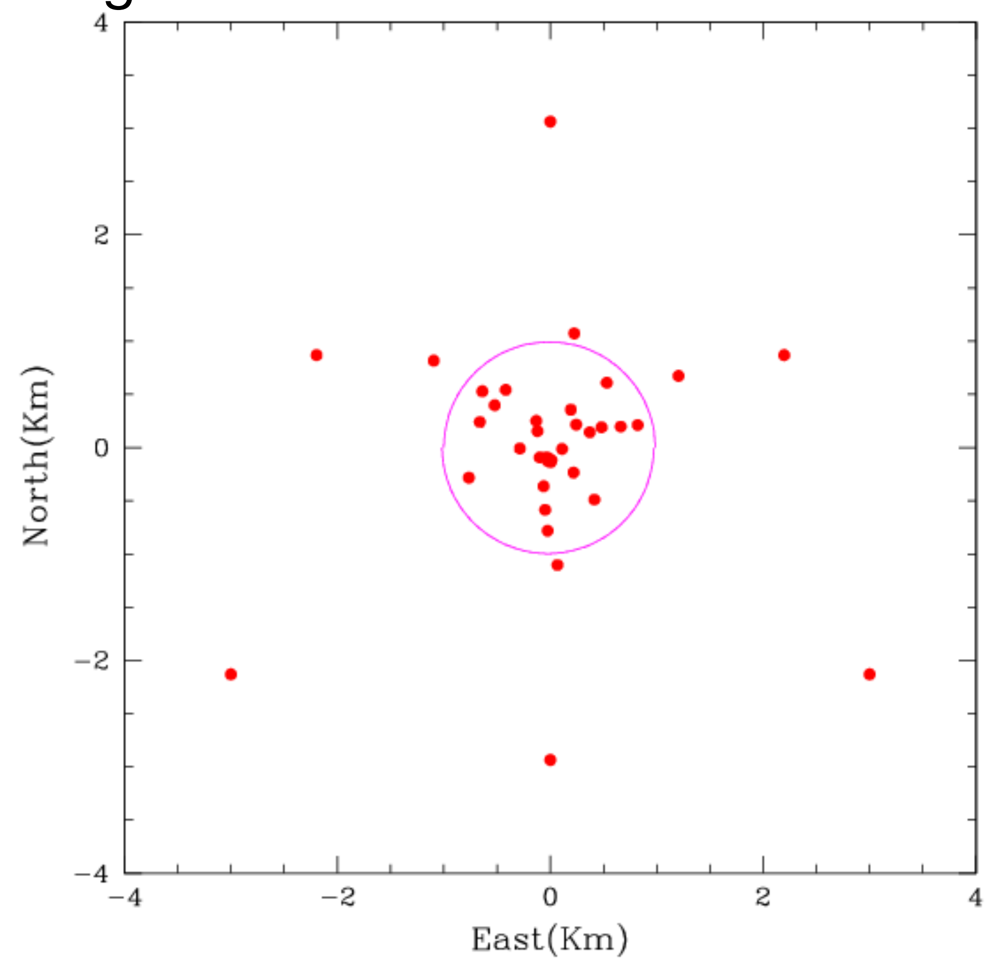
We have to dedisperse them differently

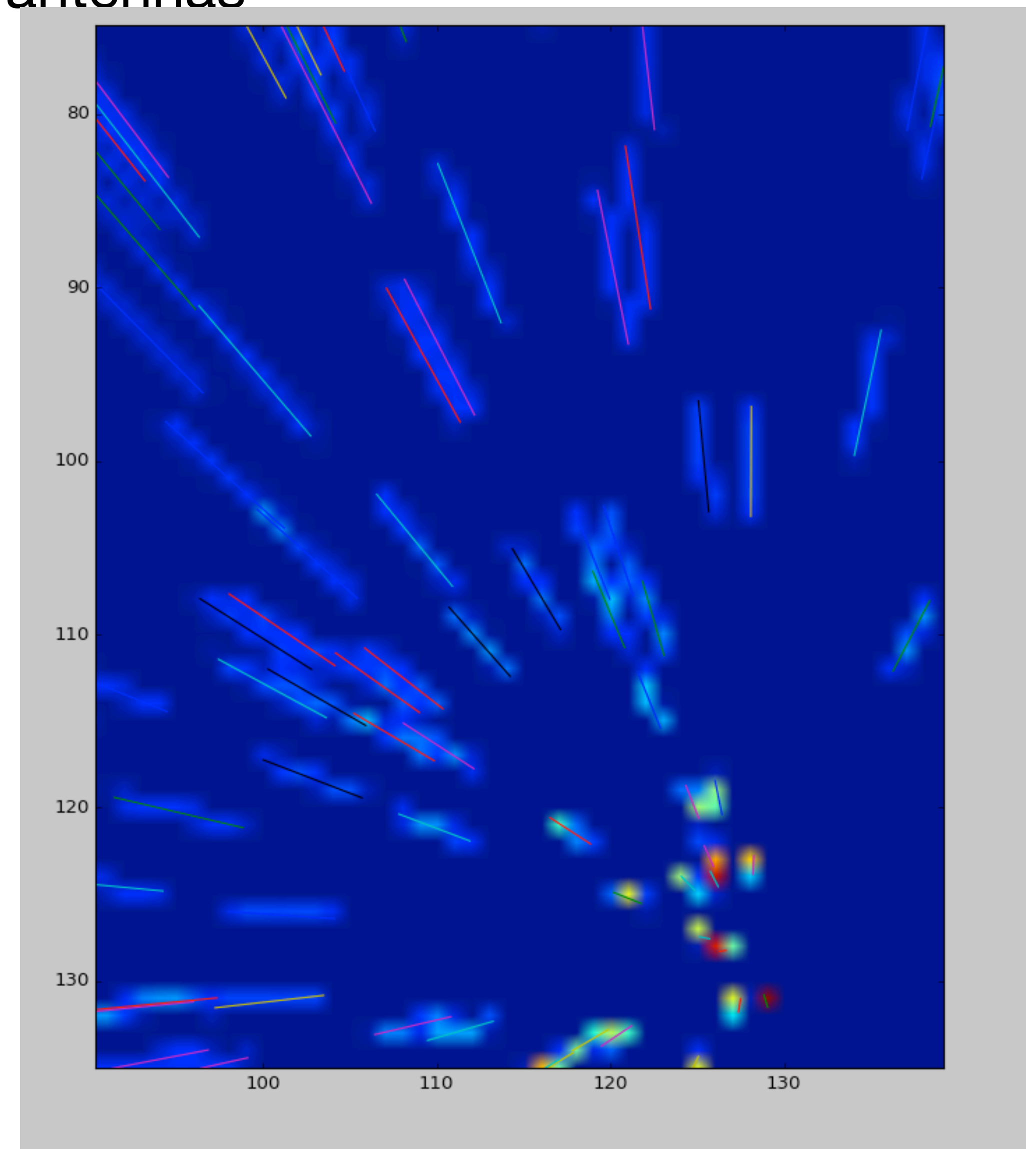


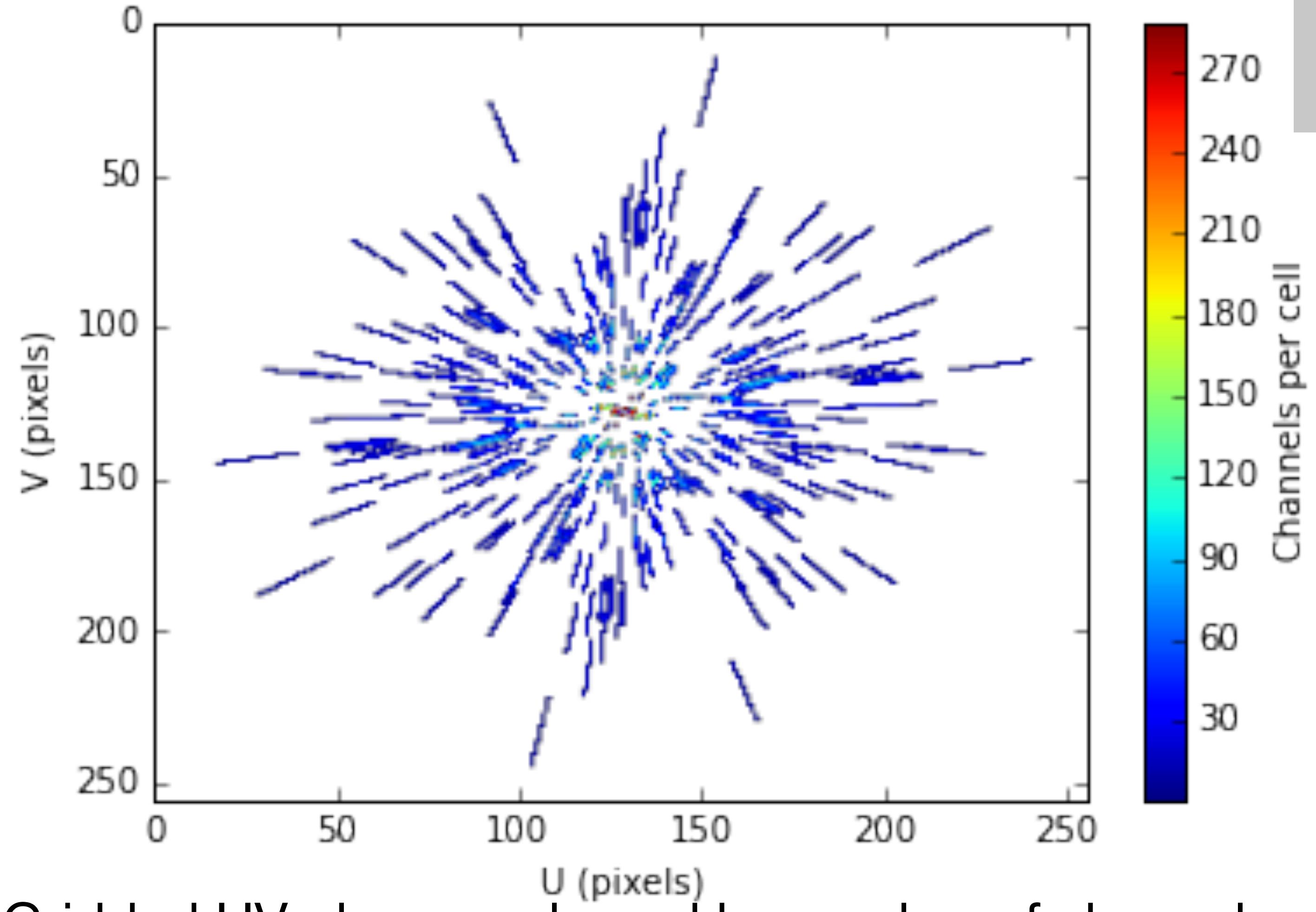
ASKAP Fast Imaging

- Typically we'll use 30 antennas within 2 km diameter - each with 288 channels, 1ms integrations.
- Discrete sampling of UV plane = Npix x Npix = 256x256 cell grid.
- The positions of the baselines are essentially static for ~30 seconds (Earth rotates slowly).
- Every millisecond we get visibilities for 30 antennas = 436 baselines x 288 channels ~ 130k measurements
- Many of the channels fall in the same cell in the grid. We'll average those (in a preprocessing step - the FDMT) by a factor of ~25x
- There are only ~5000 non-zero points in the UV plane i.e. ***the UV plane is 96% zeros (!)***

ASKAP array configuration = circle has radius=1km ~ 30 antennas

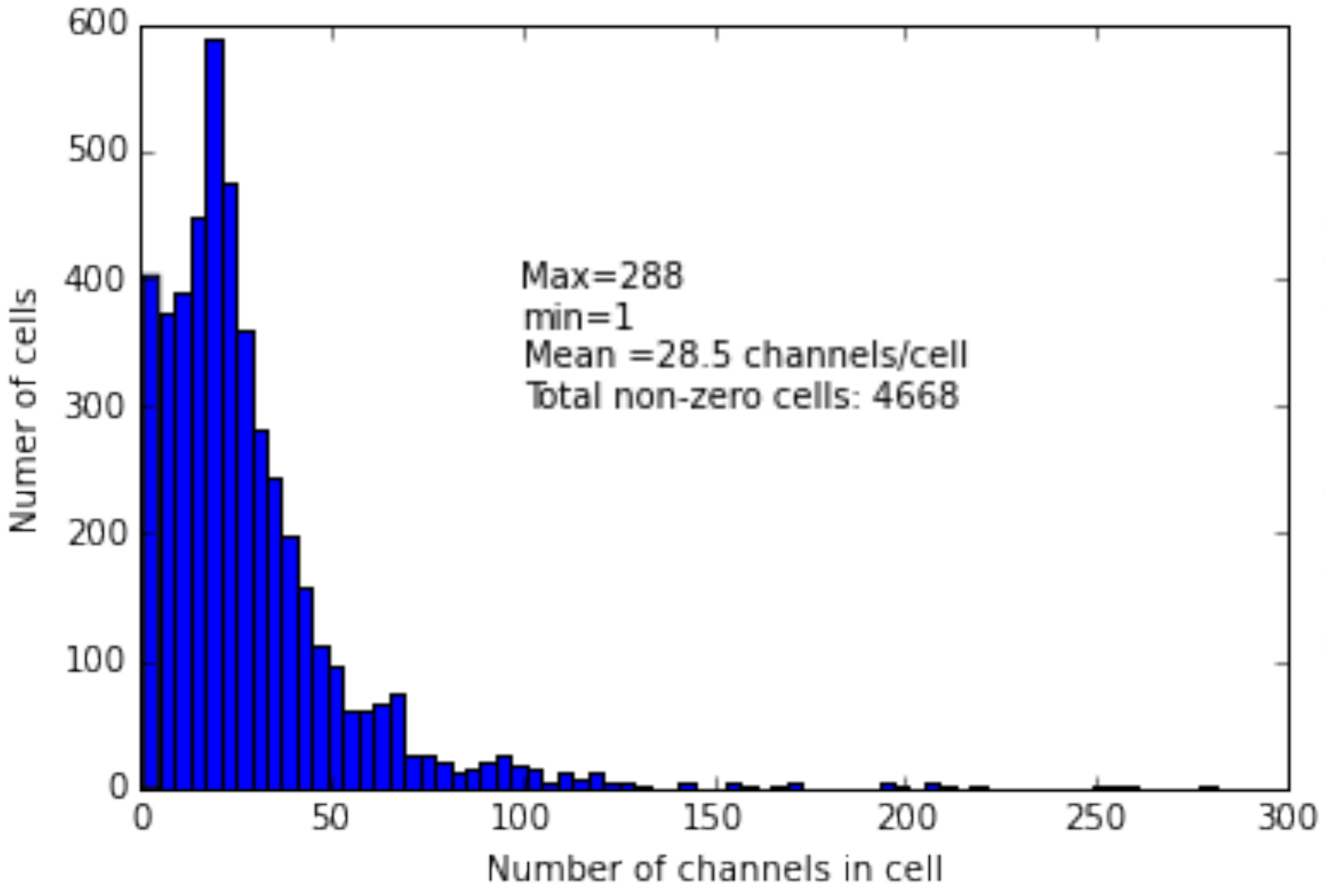




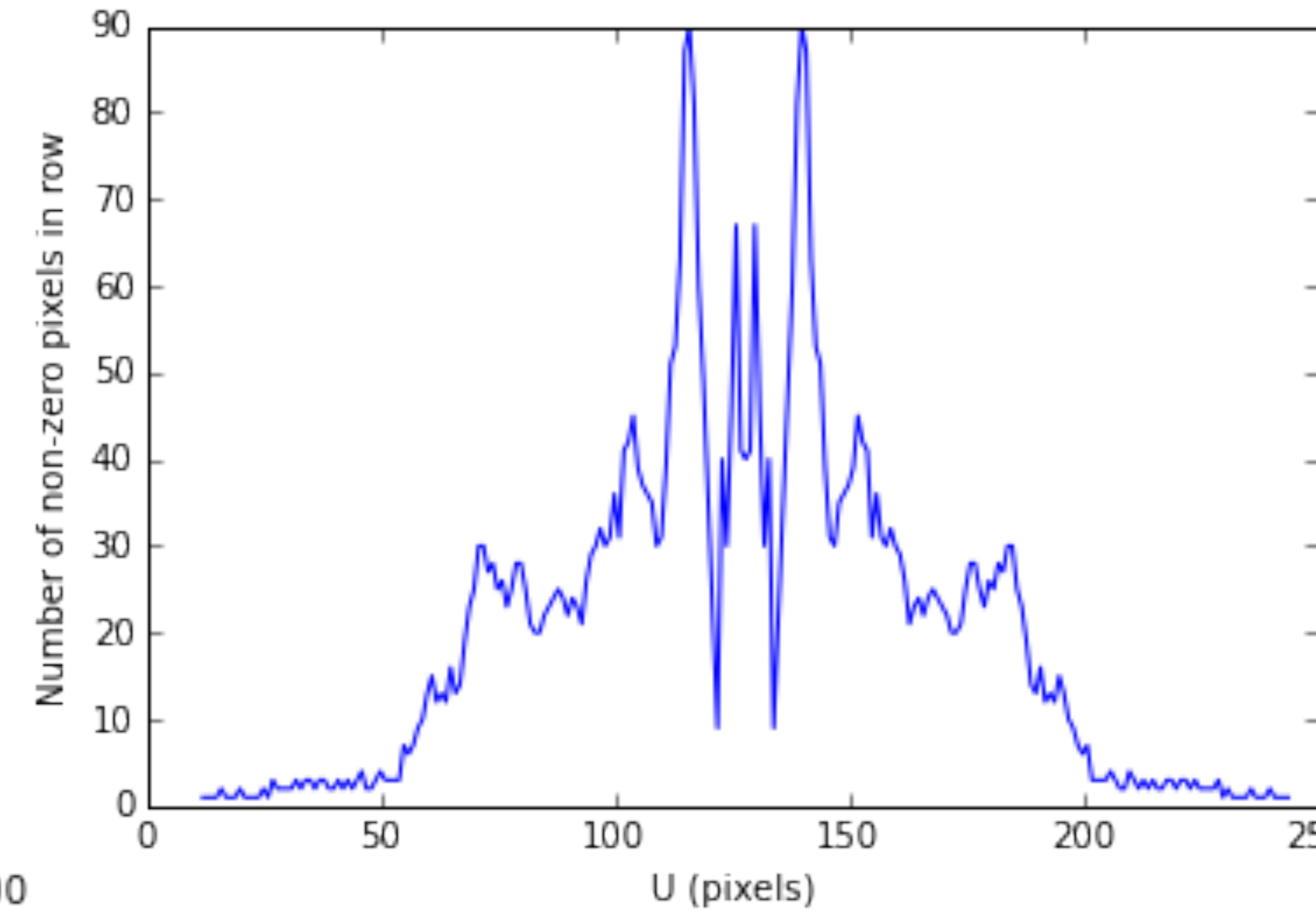


Gridded UV plane - coloured by number of channels per cell
22 antennas in 1km radius.

Statistics of sampled uv grid

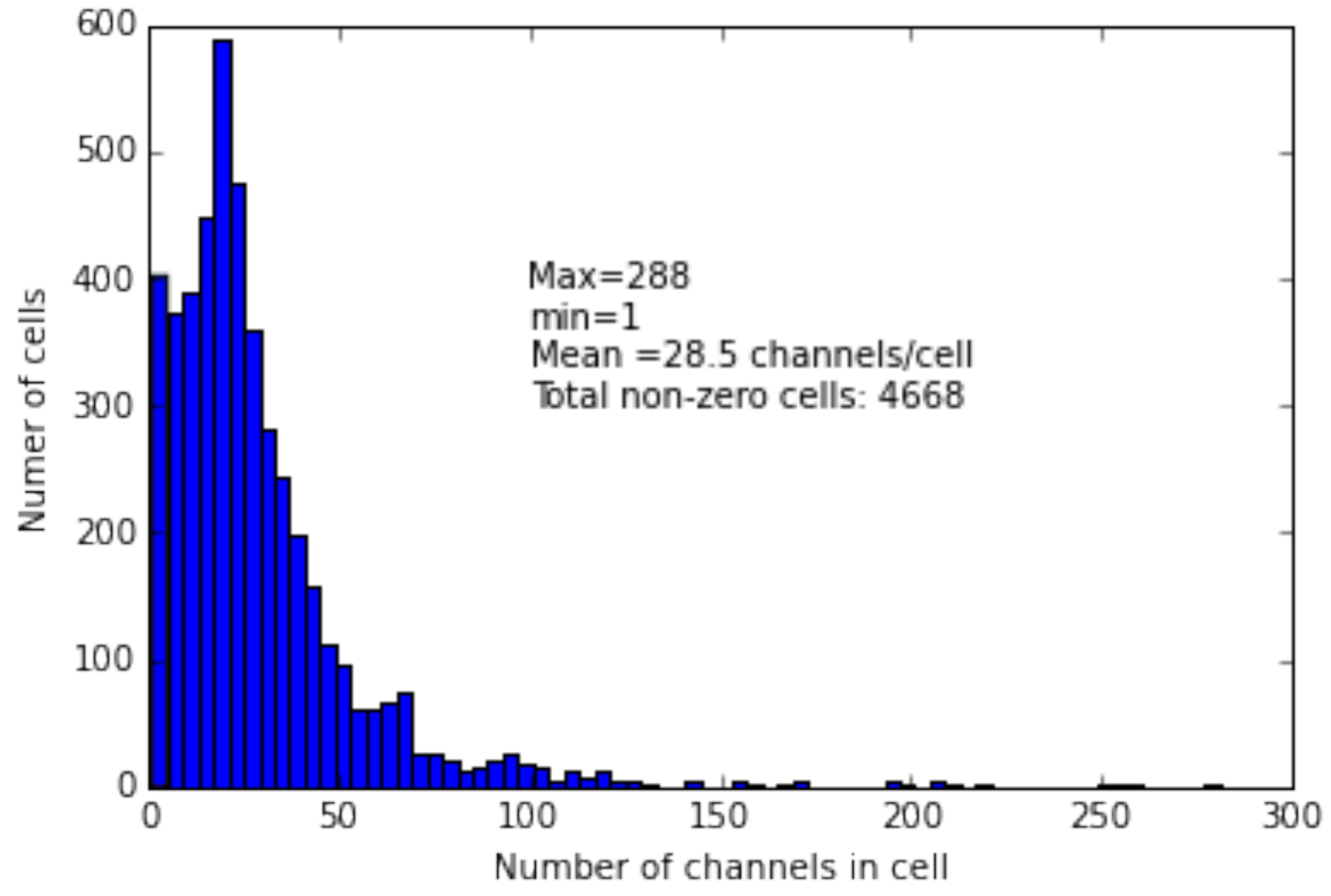


Number of channels being averaged Into each cell (this is the size of the FDMT subband)



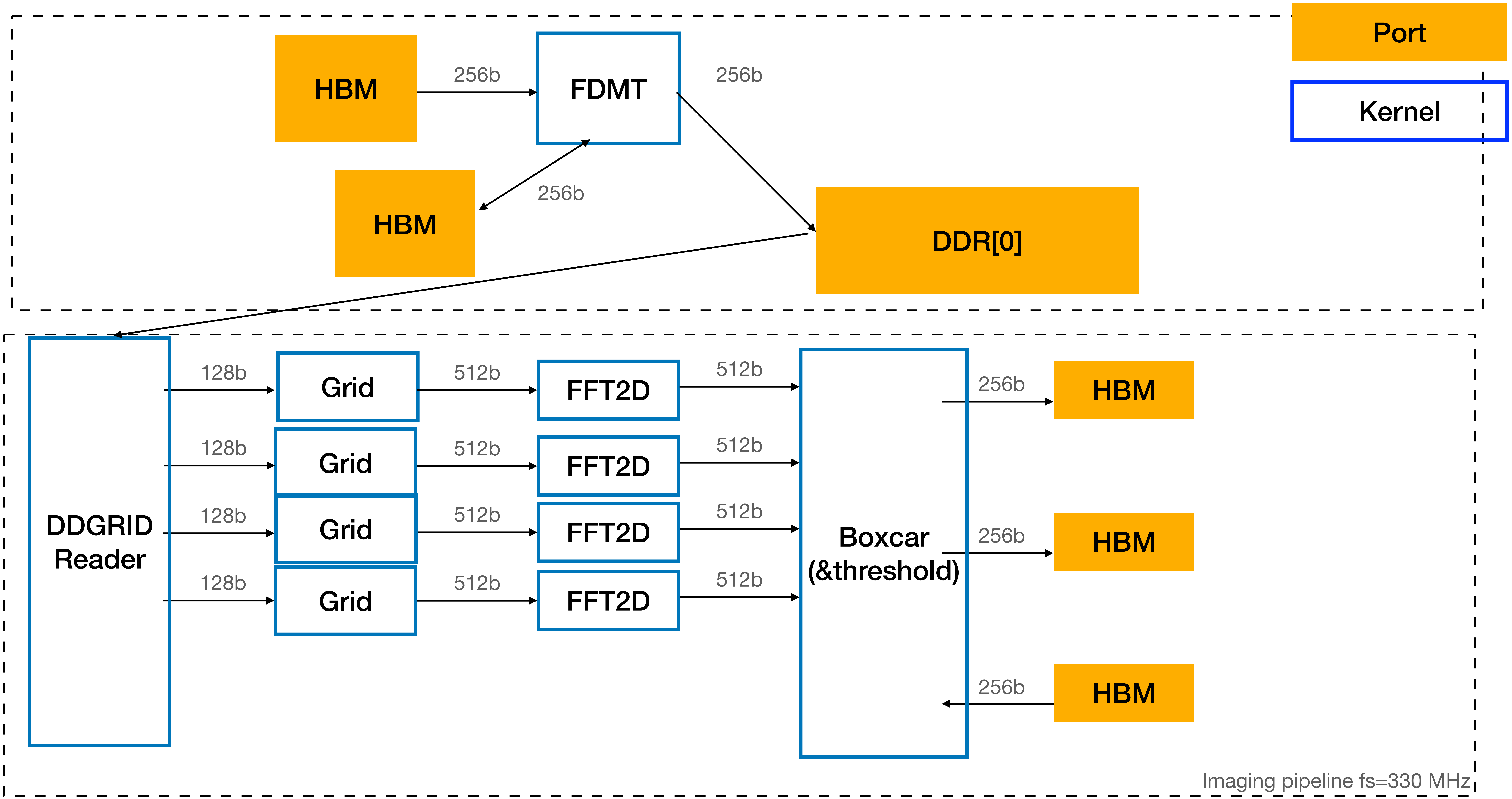
Number of non-zero values to read on a row-by-row basis - Useful for padding analysis.

2 stage dedispersion

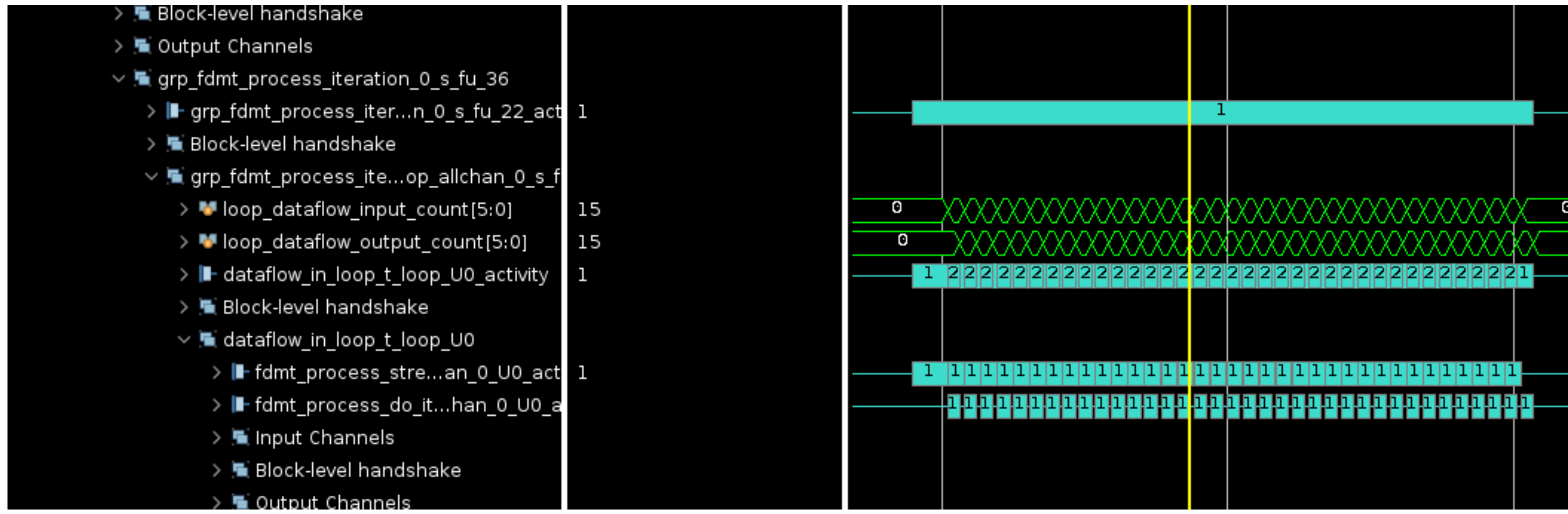


- Interferometry says: “You can’t put all the channels in a UV cell”
- Average channels/cell = 32
- If you dedispersed all 256 channels for all UVs = wasteful. You’d be dedispersing zeros a *LOT*.
- Inflate data by (NDM/NCHAN)*(NUV/NBL) = 50!
- So
- Dedisperse 32 channels at a time
- Cells with > 32 channels get split in >= two runs, and summed in the image pipeline
- Cells with <= 32 channels pad with zeros
- Inflates data by ~5
- If you want to do MAX_DM = 1024 at 888 MHz centre frequency, you need the lowest channels to be dedispsed out to 186 samples, and keep a history of 1024-186=838 of them so you can make up enough delay

Firmware architecture



Kernels: FDMT



- Computes FDMT of 8 UV cells simultaneously
- 32 channels = 5 iterations, + initialisation stage
- Produces 186 output DMs - each DM is 1 sample larger than the first. This is sufficient to let you a maximum DM of 1024 samples at the gridding stage
- Output inflated by 186 DMs / 32 channels (Inflate by 5) - final larger DMs are formed by DDGRID
- 16+16 bit arithmetic
- Delays driven by programmable lookup table (essentially the bottom frequency of the set of channels)
- In principle you can have separate frequencies for each of the 8 UV cells, but in practice DDGRID needs you to give all 8 cells the same frequency
- This gives you an awful NP-complete modified knapsack or packing optimisation problem (Dad’s working on it)
- Writes data to 16 GB DDR buffer
- Required size is 8192 UV x 186 DM x (1024 + 256) NT x 4 bytes = 7.8 GBytes - roughly half
- Memory required is proportional to MAX_DM**2 - so with this buffer (and strategy) we could get to MAX_DM=sqrt(2)*1024 = 1440 samples without changing algorithm (need to change FDMT to a 261 output rather than 186 output)

Kernels: DDGRID reader

- Processes 1 DM at a time
- Understands frequencies that the FDMT has run at
- Applies appropriate time delays to each UV
- Reads 8 samples x 8 UV cells simultaneously
- Sends to “GRID” kernels
- Programmed with lookup tables.

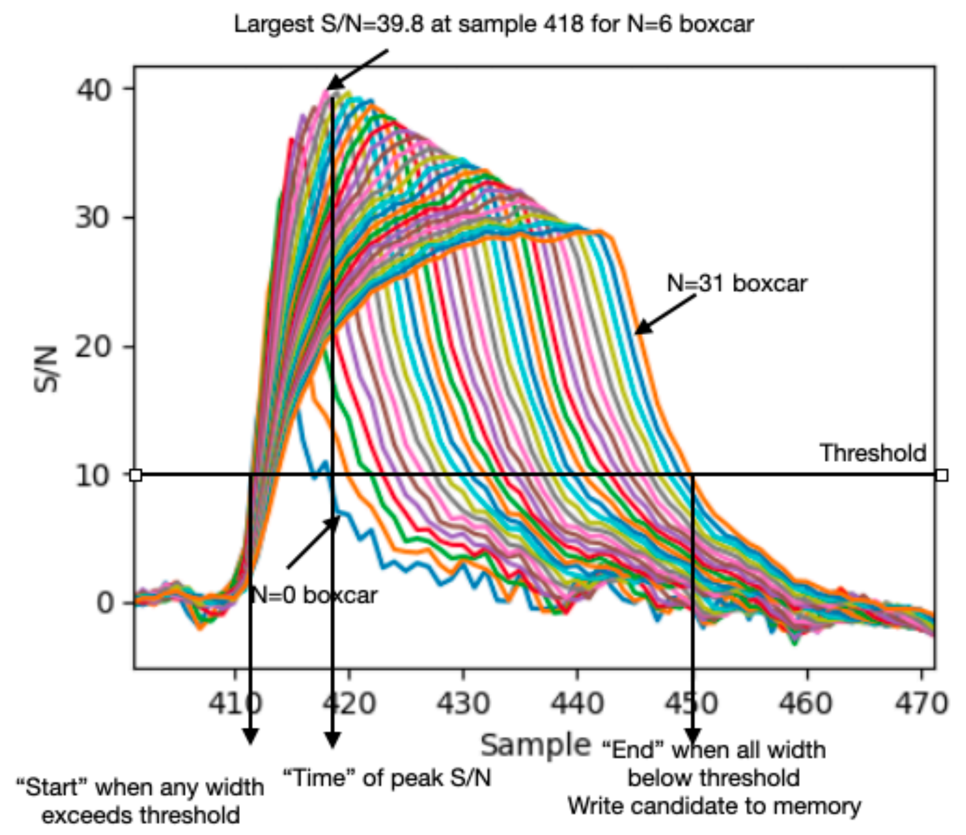
Kernels: GRID

- 4 GRID kernels drive 4 FFT kernels
- Reads 2 UV x 2 Time samples per clock
- Computes upper and lower Hermetian conjugates, so C-to-C FFT produces sequential time samples
- Writes 16 UV / clock to FFTs
- Pads with zeros in real-time
- Driven with lookup tables

Kernels: FFT

- Written specifically for us by Xilinx
- Super-sample-rate 2D FFT with SSR=16 (i.e. processes 16 pixels per clock)
- Can run back-to-back
- 4 FFTs process 8 time samples simultaneously
- Takes 4096 clocks to compute two 256x256 pixel images = 13us

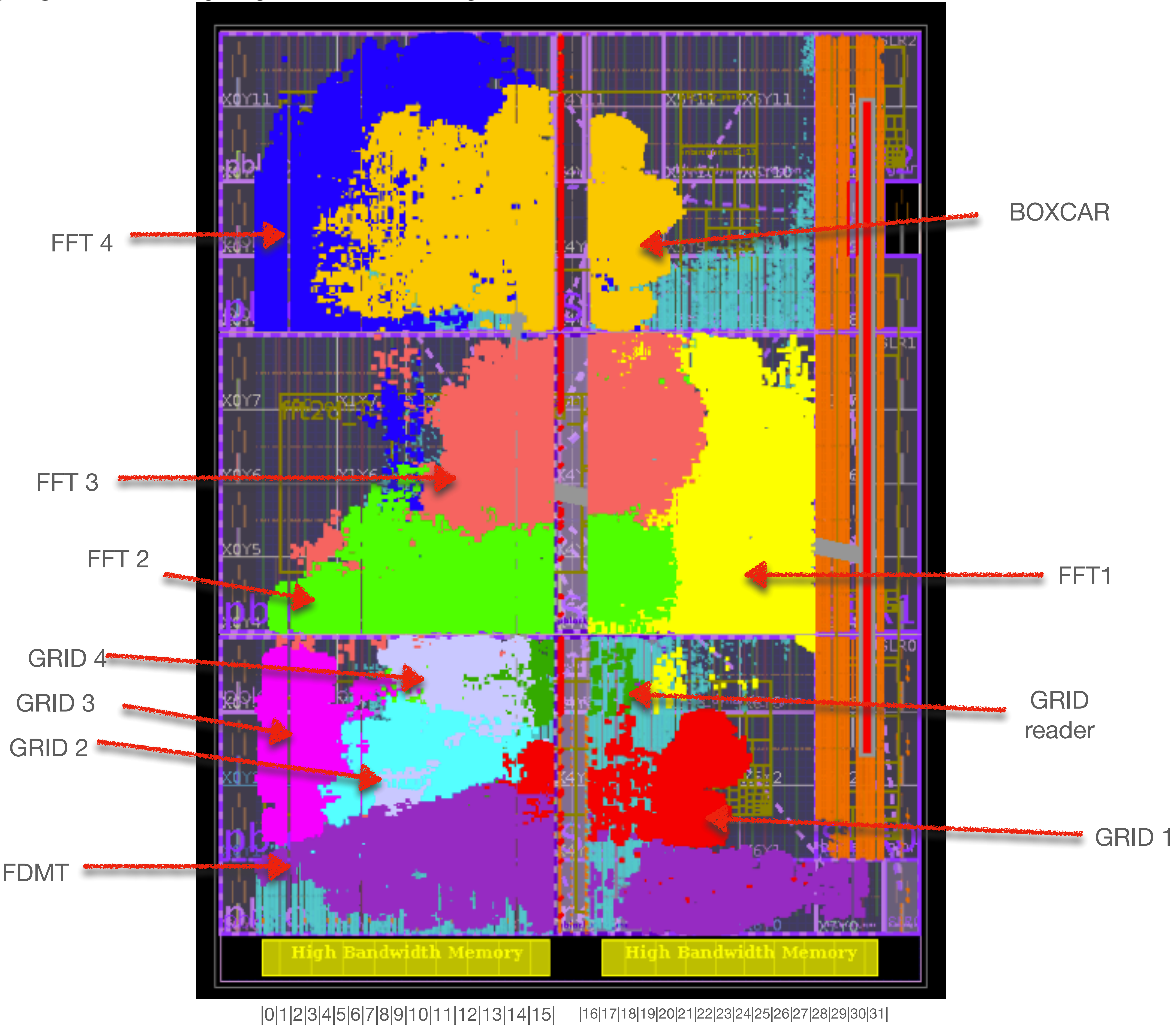
Kernels: BOXCAR



- Takes 16 pixels x 2 times x 4 FFTs and computes 8 boxcars (per clock)
- Computes maximum over **everything:**16 pixels x 8 times x 8 boxcars to give you only one result per clock.
- Compares this maximum with threshold and writes to candidate list
- A high S/N candidate will appear in many times over different pixels (due to the beam shape) and many times (due to the usual smearing etc) and many DMs (same reason).
- If you set the threshold low enough, you should be able to get quite a nice view of the shape of the FRB But the pattern will be blocky due to the way the maximum works above.
- Similar to to how FREDDA candidates work - but more of them.

What does it look like?

- fdmt_tunable_c32_1 (pfm_dynamic_fdmt_tu
- fft2d_1 (pfm_dynamic_fft2d_1_0)
- fft2d_2 (pfm_dynamic_fft2d_2_0)
- fft2d_3 (pfm_dynamic_fft2d_3_0)
- fft2d_4 (pfm_dynamic_fft2d_4_0)
- hmss_0 (pfm_dynamic_hmss_0_0)
- init_cal_combine_mss (pfm_dynamic_init_ca
- init_combine_mss (pfm_dynamic_init_combi
- interrupt_concat (pfm_dynamic_interrupt_c
- krnl_boxc_4cu_1 (pfm_dynamic_krnl_boxc_4
- krnl_grid_4cu_1 (pfm_dynamic_krnl_grid_4c
- krnl_grid_4cu_2 (pfm_dynamic_krnl_grid_4c
- krnl_grid_4cu_3 (pfm_dynamic_krnl_grid_4c
- krnl_grid_4cu_4 (pfm_dynamic_krnl_grid_4c
- krnl_sync_stream_uv_4cu_1 (pfm_dynamic_k




```
sp=fdmt_tunable_c32_1.pin:HBM[9]
sp=fdmt_tunable_c32_1.pout:DDR[0]
sp=fdmt_tunable_c32_1.histin:HBM[10:13]
sp=fdmt_tunable_c32_1.histout:HBM[10:13]
sp=fdmt_tunable_c32_1.pconfig:HBM[14]

sp=krnl_sync_stream_uv_4cu_1.in:DDR[0]

sp=krnl_grid_4cu_1.lut:HBM[0]
sp=krnl_grid_4cu_2.lut:HBM[1]
sp=krnl_grid_4cu_3.lut:HBM[2]
sp=krnl_grid_4cu_4.lut:HBM[3]

sp=krnl_boxc_4cu_1.history_in:HBM[4:7]
sp=krnl_boxc_4cu_1.history_out:HBM[4:7]
sp=krnl_boxc_4cu_1.cand:HBM[8]

slr=fdmt_tunable_c32_1:SLR0

slr=krnl_sync_stream_uv_4cu_1:SLR0

slr=krnl_grid_4cu_1:SLR0
slr=krnl_grid_4cu_2:SLR0
slr=krnl_grid_4cu_3:SLR0
slr=krnl_grid_4cu_4:SLR0

slr=fft2d_1:SLR1
slr=fft2d_2:SLR1
slr=fft2d_3:SLR1
slr=fft2d_4:SLR1

slr=krnl_boxc_4cu_1:SLR2
```

Image pipeline - real-life performance = 3440 UV

NUV=3440 - realistic NUV will be later than this so these numbers may vary

Bottlenecks

- FDMT has to write to the same memory as DDGRID is reading, at the same time. This memory can go at 20 GByte/sec bus. They need more than this total and it has overheads and is hard to get right.
- FDMT needs to write NUV x NDOUT x 4 bytes / 1.7ms = 3.5 GByte/sec
- DDGRID eneds to read NUV x NDM x 4 bytes / 1.7ms = 19.7 Gbyte/sec
- Even if DDGRID had the bus to itself (which it doesn't) the FFTs can only consume data at a certain rate. I.e. 1 FFT takes 4096 clks x 3.3ns = 13.5us
- FFT processing back-to-back is 13.5us x NDM x NT/8 = 442 ms.

What will we take to first light?

- Correlator firmware
    - 162kHz channels
    - Stokes I
    - Selectable baselines
    - Configurable dump time = 0.432, 0.864, 1.7, 3.4ms.
    - 32bit x2 output (possibly 16xbit x 2).
- FSCRUNCH to 1MHz in software - configurable DM in FSCRUNC
- 20 beams
- 256x256 pixel images
- Max DM = 1024 samples
- 8 linear-spaced boxcars: 1,2,3,4,…8
- 500 DM trials (TBD)
- ~4000 UV cells ~= 24 antennas (TBD)
- Quite a large trade space to play in, e.g.
    - UV cells vs DM trials vs angular resolution vs number of antennas/baselines

Tradeoffs we can do now

- We have to work around the bottlenecks and ideally keep the FFTs running all the time.
- Adding bandwidth makes life harder with only sqrt(NCHAN) improvement in sensitivity
- Adding antennas makes life harder but with NUV improvement in sensitivity
- See planning tool. (craco_plan.py)

CRACO needs you!

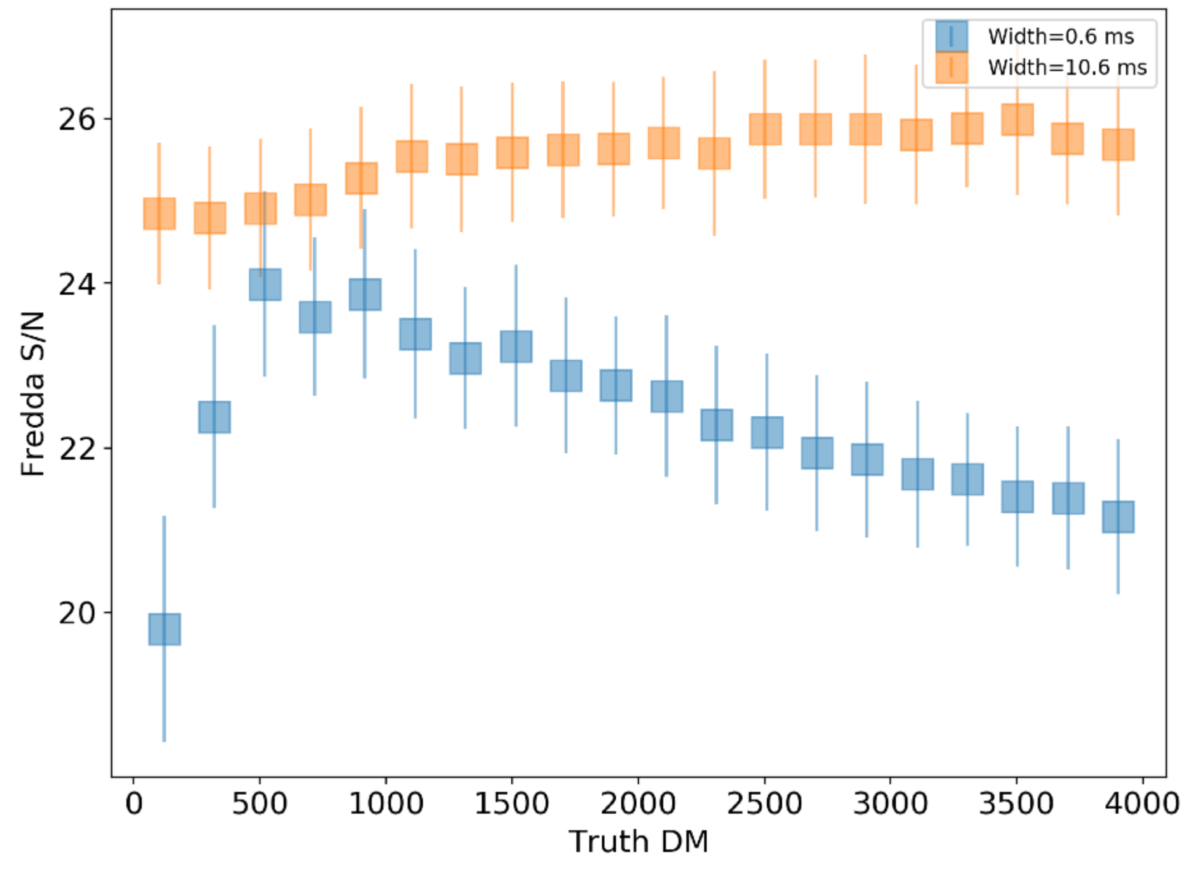
- Improvements to planning tool - program up different tradeoffs
- General inspection / debugging tools
- Calibration and RFI flagging loop
- Checking UV data / metadata (UV coords)
- Candidate grouping / thresholding / flagging / classifying
- Archiving
- NP-complete packing problems
- Subtle ways of computing lookup tables for better S/N
- Requests coming soon!

Scope for upgrading!

- Huge amount of improvements that can be made. Each has its complexities.
- I’m confident we can improve things a lot once we have a working system, e.g.
- Dedispersing at 162 kHz (nice thing here: The FDMT output rate doesn’t change).
- More DM trials (up to 1000)
- Larger MAX DM
  - Don’t compute large DMs for higher frequencies
  - Discard low DMs from history when they’ve been computed
- Upgrade cards to U55C
- More beams (with more hardware).

Improve Sensitivity to Narrow pulses

- Half-sample offsets for unresolved pulses reduce sensitivity - but this can be recovered during dedispersion
- Search over offset improves S/N by sqrt(2) (best case) for narrow pulses with DM < diagonal (300 pc/cm3)



Harry Qiu