Fast Imaging

for Slow Transients

Anna Scaife

2016/04/13

Jodrell Bank Centre for Astrophysics
University of Manchester

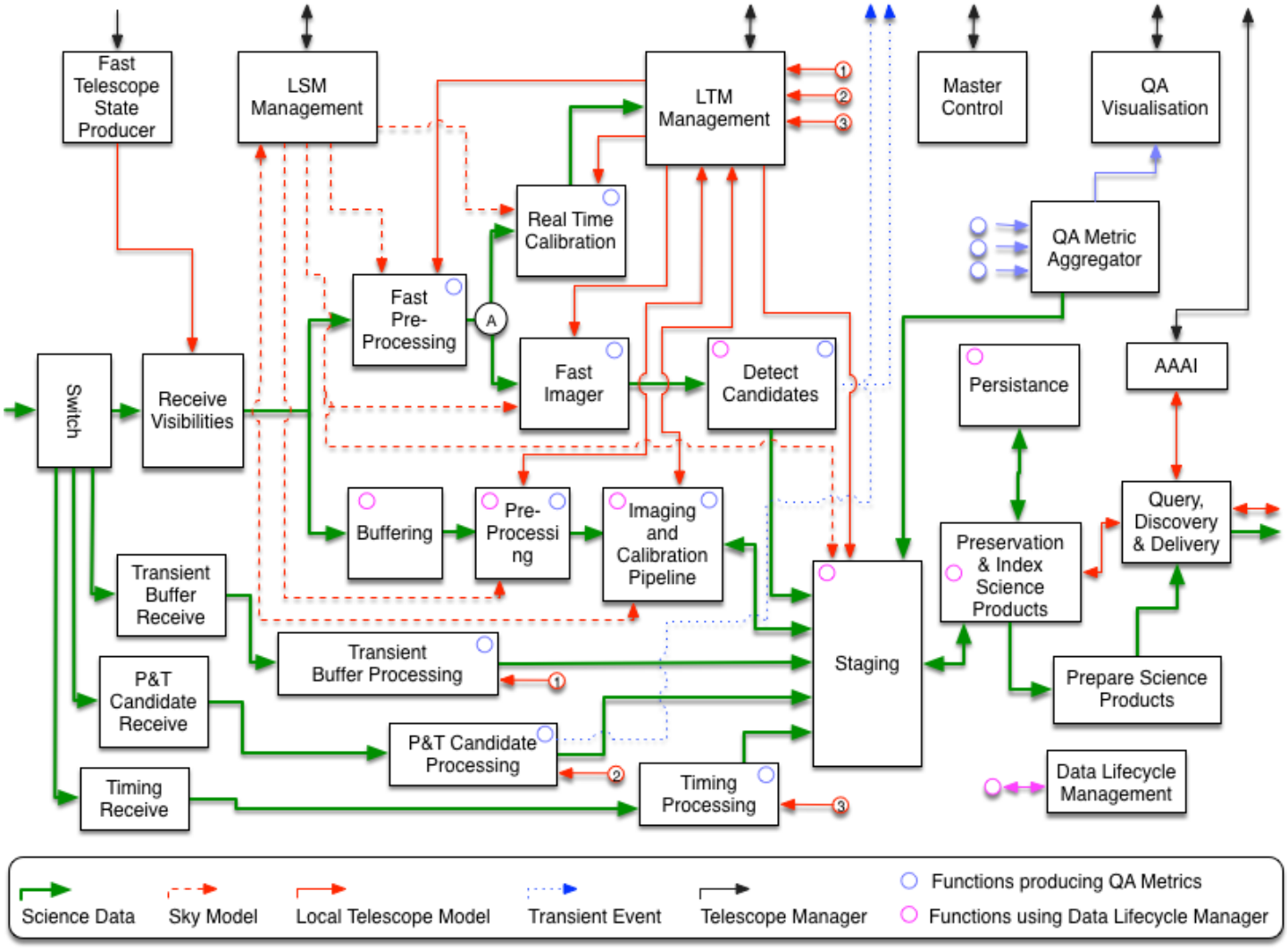
What are SKA slow transients?

Slow Transients

- Are transients that can be detected in the image plane, i.e. still significant in snapshot images.
- Transients are rare in terms of no. per steradian, therefore imaging a large FOV is preferable.
- Transients are expected to produce triggers/alerts for other instruments – pipeline must run as close to real time as possible.
- Triggers are used to initiate other observations – localisation is important.

The SDP Fast Imaging Pipeline

Context


Fast Telescope State Producer
LSM Management
LTM Management
Master Control
QA Visualisation
Real Time Calibration
QA Metric Aggregator
Fast Pre-Processing
A
Switch
Receive Visibilities
Fast Imager
Detect Candidates
Persistance
AAAI
Buffering
Pre-Processing
Imaging and Calibration Pipeline
Query, Discovery & Delivery
Transient Buffer Receive
Preservation & Index Science Products
Transient Buffer Processing
Staging
P&T Candidate Receive
Prepare Science Products
P&T Candidate Processing
Timing Receive
Timing Processing
Data Lifecycle Management
Science Data
Sky Model
Local Telescope Model
Transient Event
Telescope Manager
Functions producing QA Metrics
Functions using Data Lifecycle Manager

Context

The outputs from the SDP Fast Imager are:

- A catalogue of variable sources (update to GSM)
- VO alerts in ~ real time

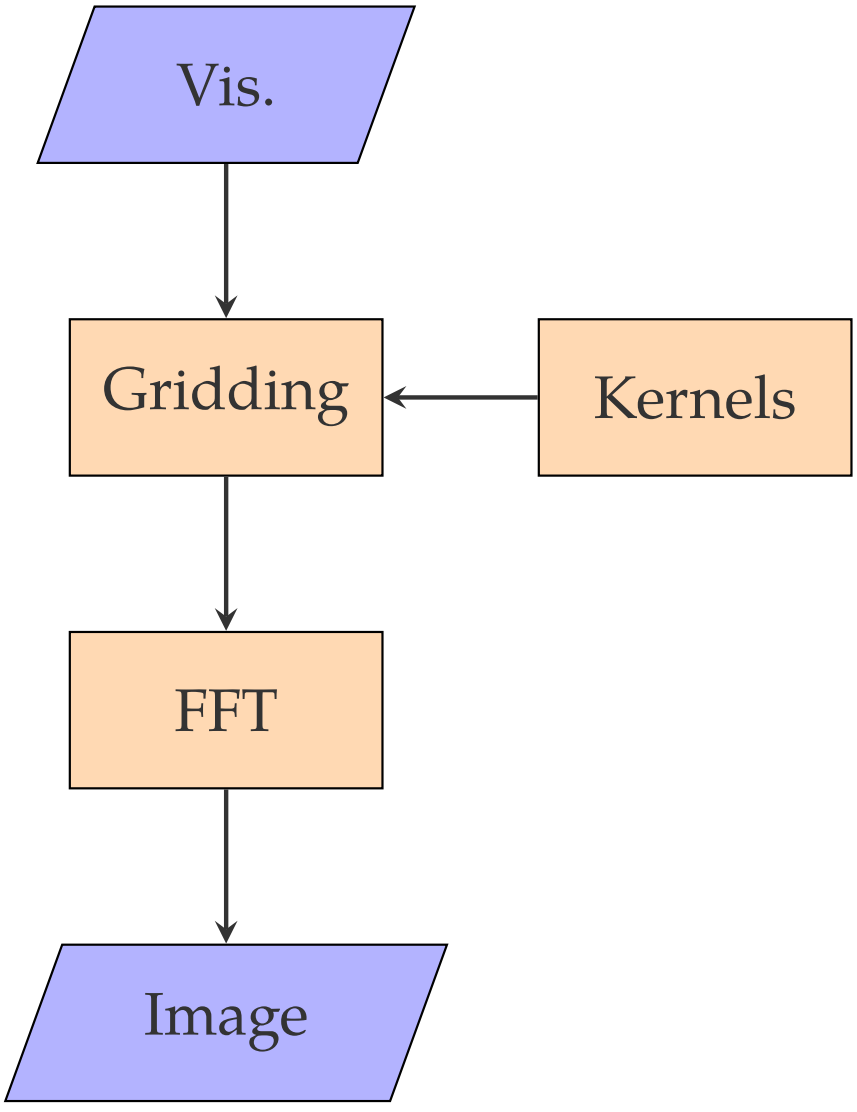
Functional Flow

- The two major steps of the fast imaging pipeline are the same as any imaging pipeline (probably).
- The visibilities that will be gridded are differenced with the Global Sky Model (GSM).
- Transient sources are expected to be rare, therefore this differenced sky will be sparse.

Overview

- Generating images from radio data requires a very large number of simple, repetitive calculations.
- The run time scales with either the number of visibilities (gridding, phase rotation) or the image size in pixels (FFT, cleaning).

Functional Flow - Continuum



Example:

1 Hour observation

For continuum imaging, visibilities are buffered over the duration of the observation and then gridded together.

The FFT is done (effectively) once.

Functional Flow - Fast

Example:

1 Hour observation

For fast imaging, visibilities stream continuously into the pipeline at a snapshot cadence of ~1 s

Overall the same number of visibilities are gridded.

The FFT is done 3600 times.

Prototyping

FFT

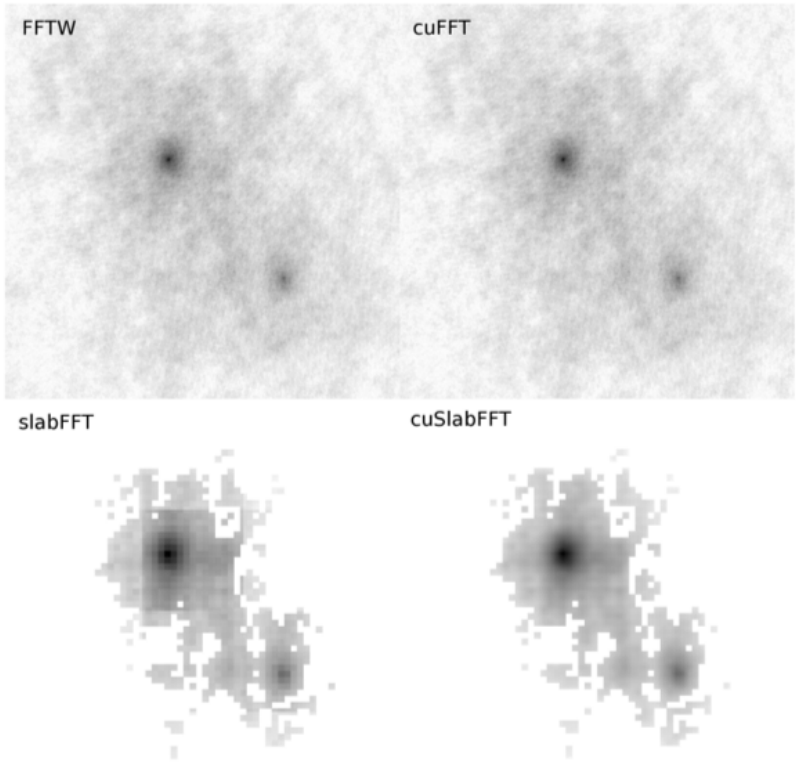
Originally we thought that the FFT would be the limiting factor for fast imaging.

We looked at non-traditional alternatives, like the Sparse FFT algorithm.

Sparse FFT

5.4 s

1.9 s

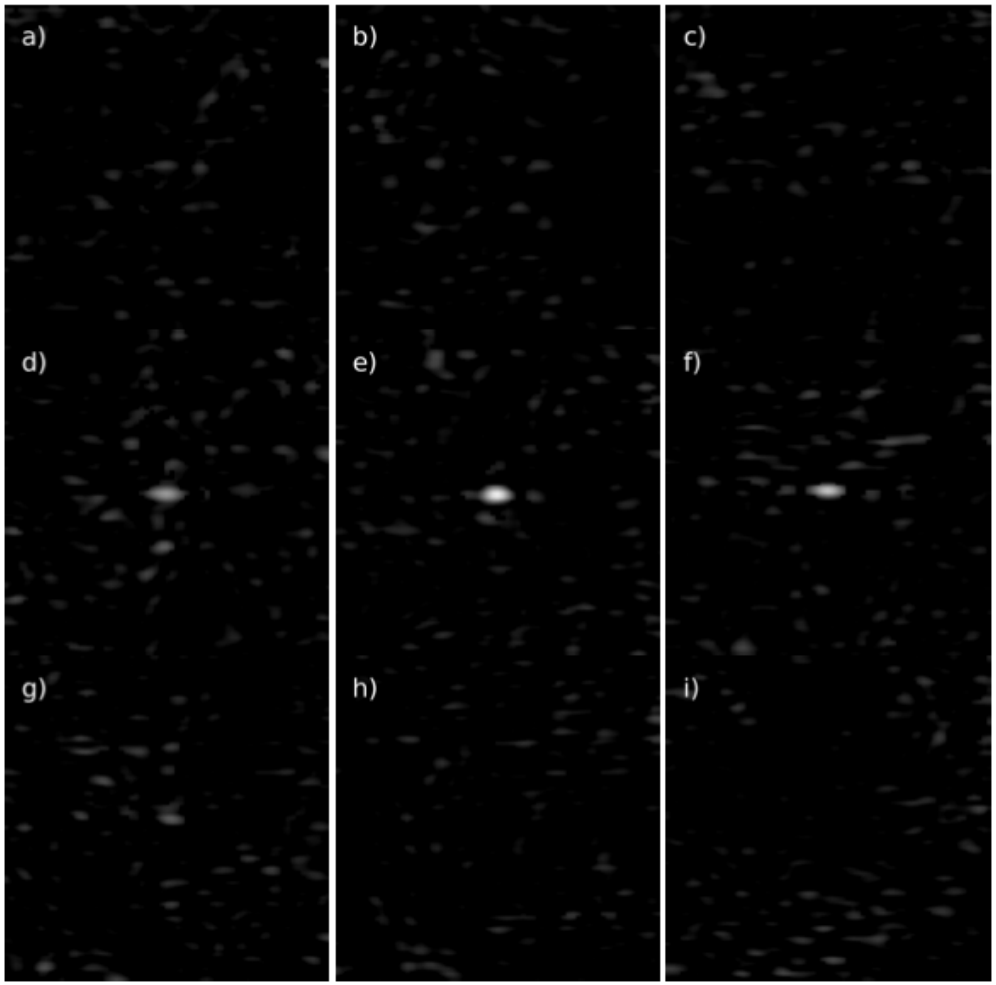


240 ms

36 ms

2000 × 2000 cut-outs of 16384 × 16384 images

Sparse FFT



JVLA data 5 ms snapshots (courtesy of Casey Law)

Sparse FFT

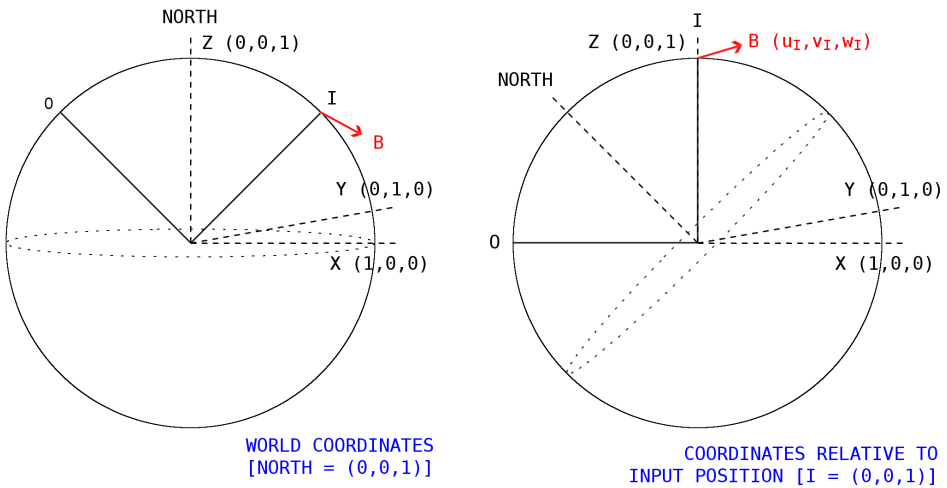
Likelihood is that we won't need the Sparse FFT because facetting will make the images small.

The facets will probably be necessary for local phase calibration in the fast pipeline - otherwise the GSM differencing won't work effectively.

Proto-typing: Phase Rotation

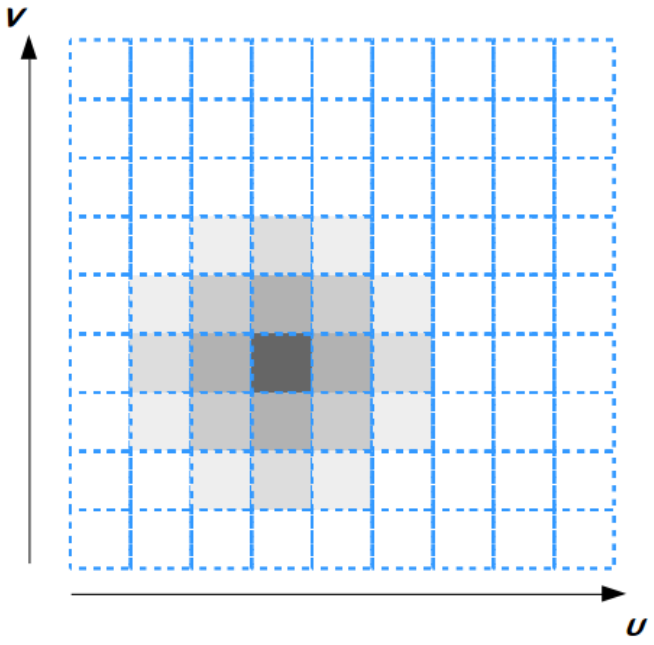
Change the phase centre to a new phase position on the sky.
Two steps:

- Rotate the u, v, w coordinates about the celestial sphere.
- Apply a phase shift to the complex visibilities.



HIGHLY PARALLELIZABLE

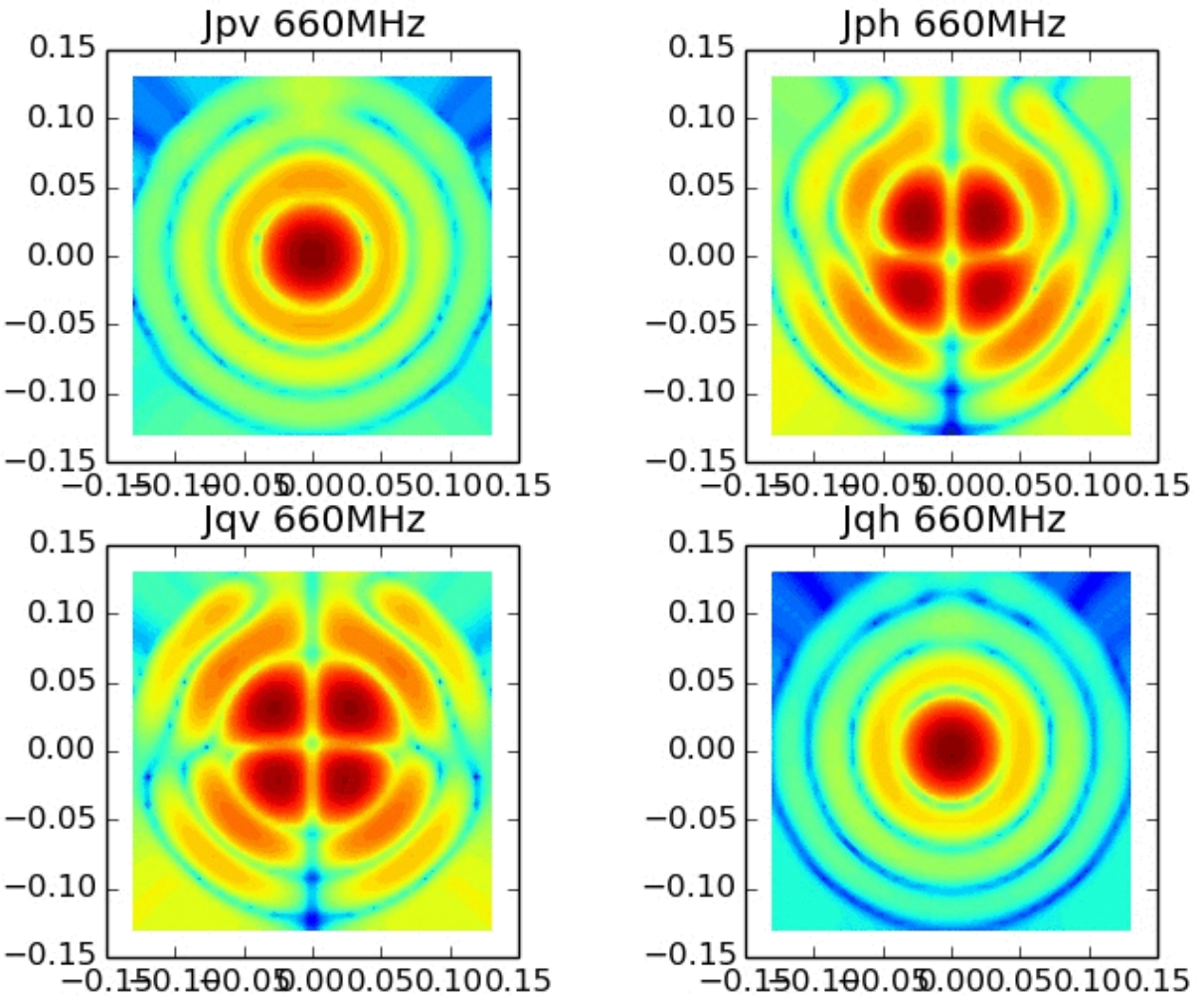
Proto-typing: Gridding



- Each visibility is placed onto the 'grid', using a delta function at the uv grid position convolved with an $N \times N$ 'kernel'.
- The kernel performs functions such as w-projection, anti-aliasing and A-projection.
- Total number of operations = $N_{\text{samples}} \times N_{\text{channels}} \times N_{\text{kernel}}^2$.

HIGHLY PARALLELIZABLE

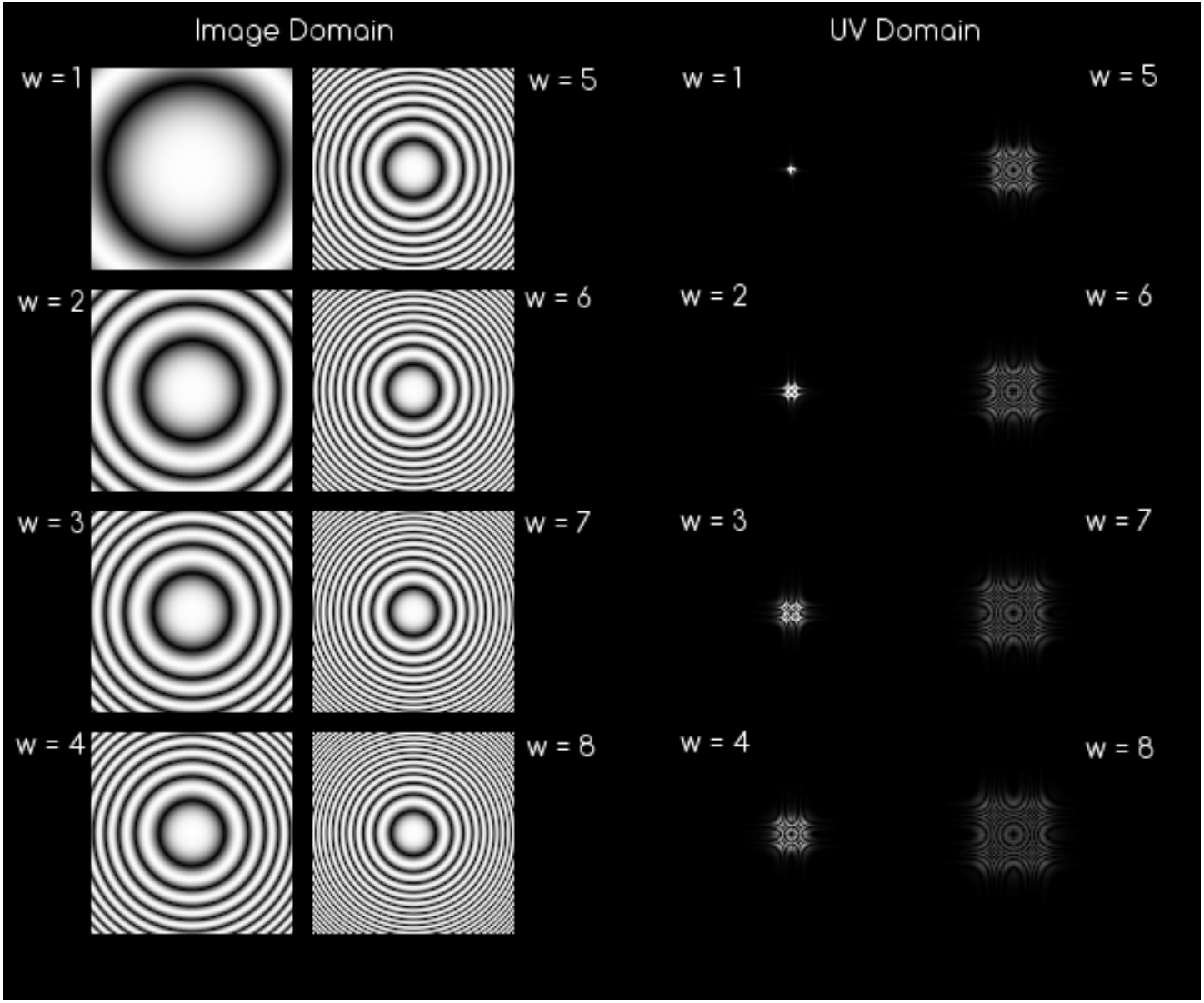
A-kernel


Jpv 660MHz
Jph 660MHz
Jqv 660MHz
Jqh 660MHz
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.15−0.10−0.05 0.00 0.05 0.10 0.15

Proto-typing: w-Kernel

- 2D complex exponential function, required to correct for non- coplanar baselines (i.e. visibilities with non-zero w-coordinates)
- The larger the w-value, the larger the kernel.

Proto-typing: w-Kernel

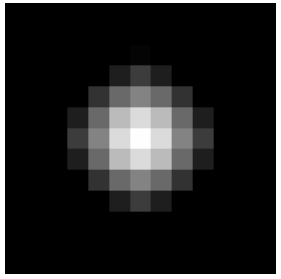


Proto-typing: Anti-aliasing Kernel

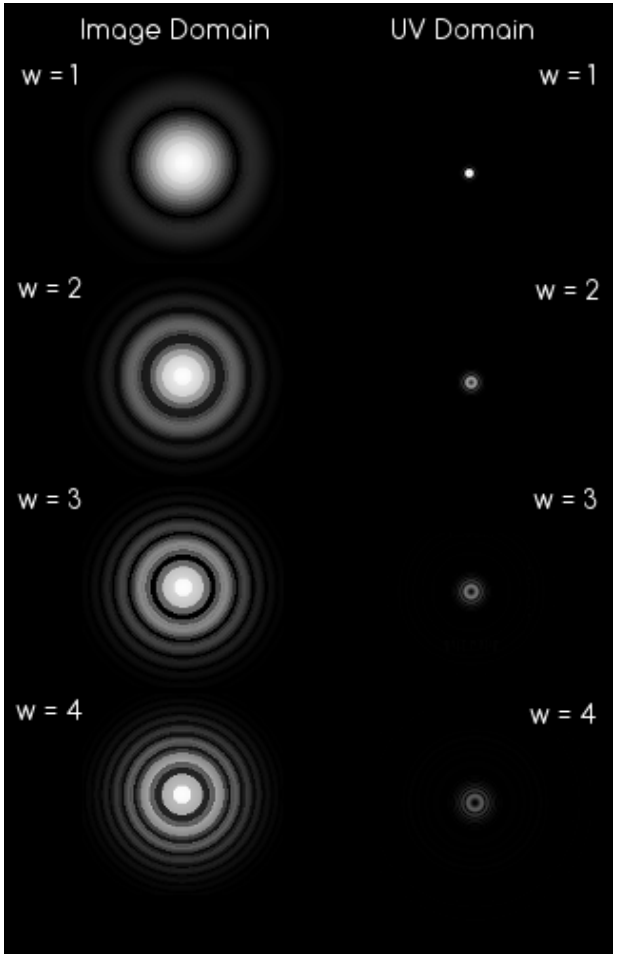
- Prolate spheroidal function, required to prevent aliasing effects due to finite resolution of uv grid.
- Generated in the uv domain, and usually 7×7 pixels.

Convolution introduced in the uv plane during gridding, and later removed from the image plane by dividing the dirty image by the Fourier transform of the kernel.

$$V * K = I \times \tilde{K}$$



Proto-typing: Combined Kernel



- The anti-aliasing and w-kernels are multiplied together in the image (lm) domain, and FFT'd into the uv domain.
- Equivalent to a convolution in the uv domain.
- Sky image must be divided to remove the anti-aliasing kernel.
- w-kernel does not need to be compensated for.

Pipelines, Pitfalls & Progress

Demonstrator

- Algorithms are ideally suited to a parallel architecture, such as Graphics Processing Units (GPUs).
- We (Chris Skipper) have constructed a GPU imager from various SDP prototyping code – phase rotation, gridding, image-plane reprojection, etc.

GPU Prototyping

- Originally developed for graphics cards – games, animation, CAD, etc.
- Designed to perform simple mathematical operations with very high efficiency.
- Highly suited to performing scientific tasks.

Figure: GPU Card

GPUs can contain many thousands of cores, but these are really just arithmetic and logic units (ALUs) – much simpler than CPU cores.

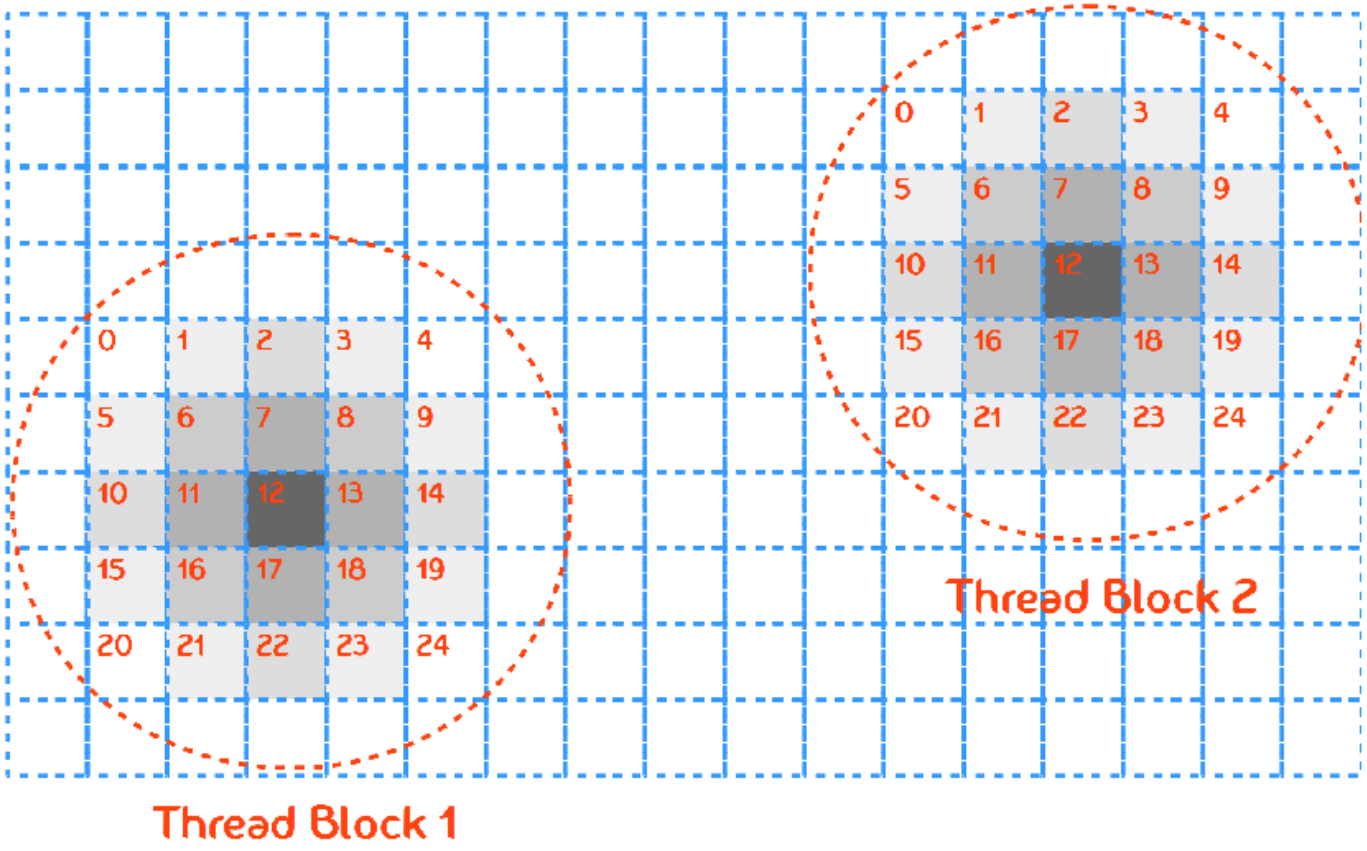
GPU Prototyping

Figure: Tesla K40

- 2,880 CUDA Cores
- 12 GB GPU RAM
- 235 W Power Consumption
- Cooled by passive heat sink
- CUDA Compute Capability 3.5
- PCI-e Interface

- Max 1,024 threads per block
- 64 k Shared Mem / m'processor

Scatter Gridding



Kernels held as 2D texture maps

ALMA Test Simulation

`ALMA Test Simulation: 86,400 visibilities`

`kernel size = 61 × 61`

	CPU [ms]	GPU [ms]	Speed-up
Gridding	13,337	41	330
FFT	228	2	110
(Hogbom Clean)	228	125	1.8

Kernel Generation

KERNEL GENERATION: These numbers do not include kernel generation - generating kernels for 128 w-planes takes a further 3 s (GPU).

Conclusions

Conclusions

- SKA Fast Imaging will produce a ~ real time catalogue of variable sources
- SKA Fast Imaging will automatically produce alerts when variability is detected
- Fast Imaging will run at a cadence of approx. 1 s
- An end-to-end prototype is being built and tested on real data