Fast Sparse 2-D DFT Computation using Sparse-Graph Alias Codes

Frank Ong, Sameer Pawar and Kannan Ramchandran

Berkeley
UNIVERSITY OF CALIFORNIA

Fourier Imaging

- Many imaging modalities acquire in the Fourier Domain:
  - Magnetic Resonance Imaging (MRI)
  - Computed Tomography (CT)
  - Fourier Optics
  - Astronomical Imaging
- Want to take few Fourier samples to reduce time and costs

Compressed Sensing

- Exploits sparsity to go beyond Nyquist rate
- Provides good reconstructed images

- Most algorithms alternate between Fourier and image domain
    - >100 FFTs! >10-minute reconstruction!

This Work: 2D-FFAST
(Fast-Fourier-Aliasing-based-Sparse-Transform)

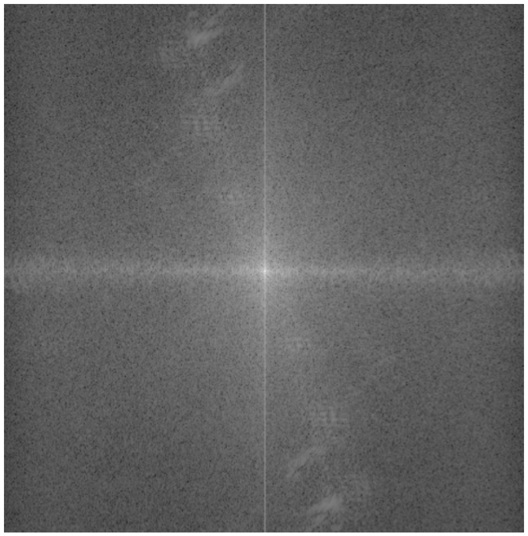
- Goal: Fast in both acquisition and reconstruction
  - Real-time reconstruction

- Many previous works in sparse FFT:
  - Gilbert et al. 2002
  - Indyk et al. 2012
  - Hassanieh et al. 2014
  - Iwen 2010
  - And many more…
  - Mostly 1D results

- This work:
  - Generalizes 1D-FFAST framework of Pawar & Ramchandran 2013
  - Illustrate 3 key ideas

Aliasing with Dense Spectrum

2D Signal

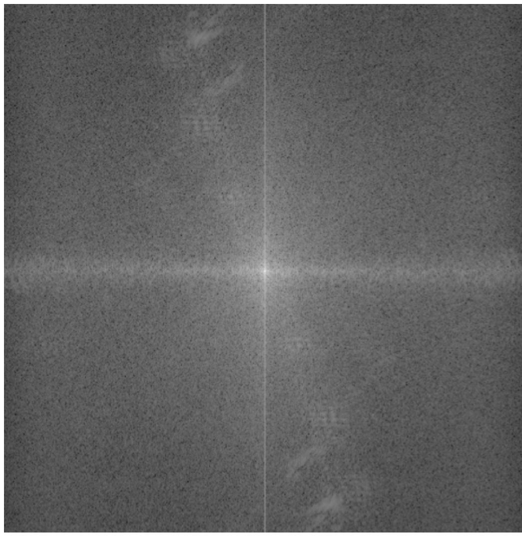


2D Spectrum

Aliasing with Dense Spectrum

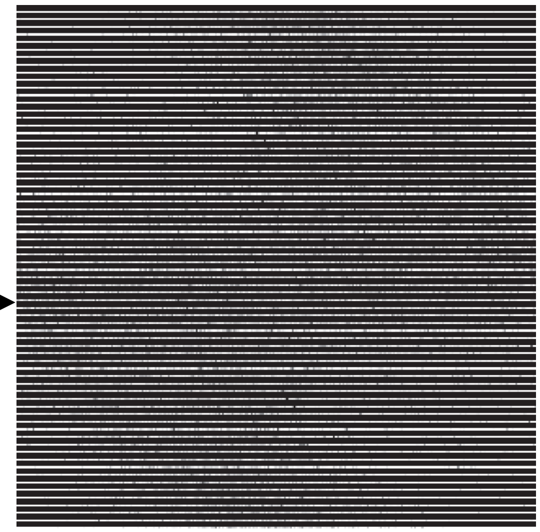
- Everything gets aliased on top of each other

2D Signal



Subsampling

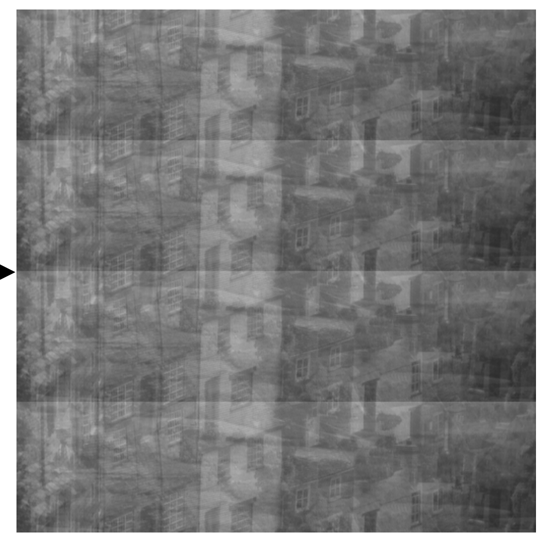
Subsampled 2D Signal



2D Spectrum

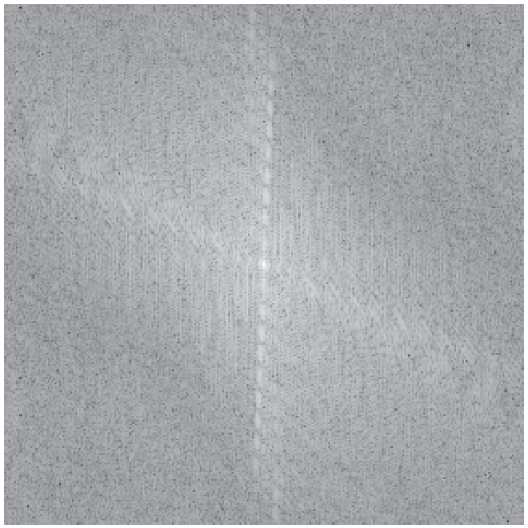
Aliasing

Aliased 2D Spectrum



Aliasing with Sparse Spectrum

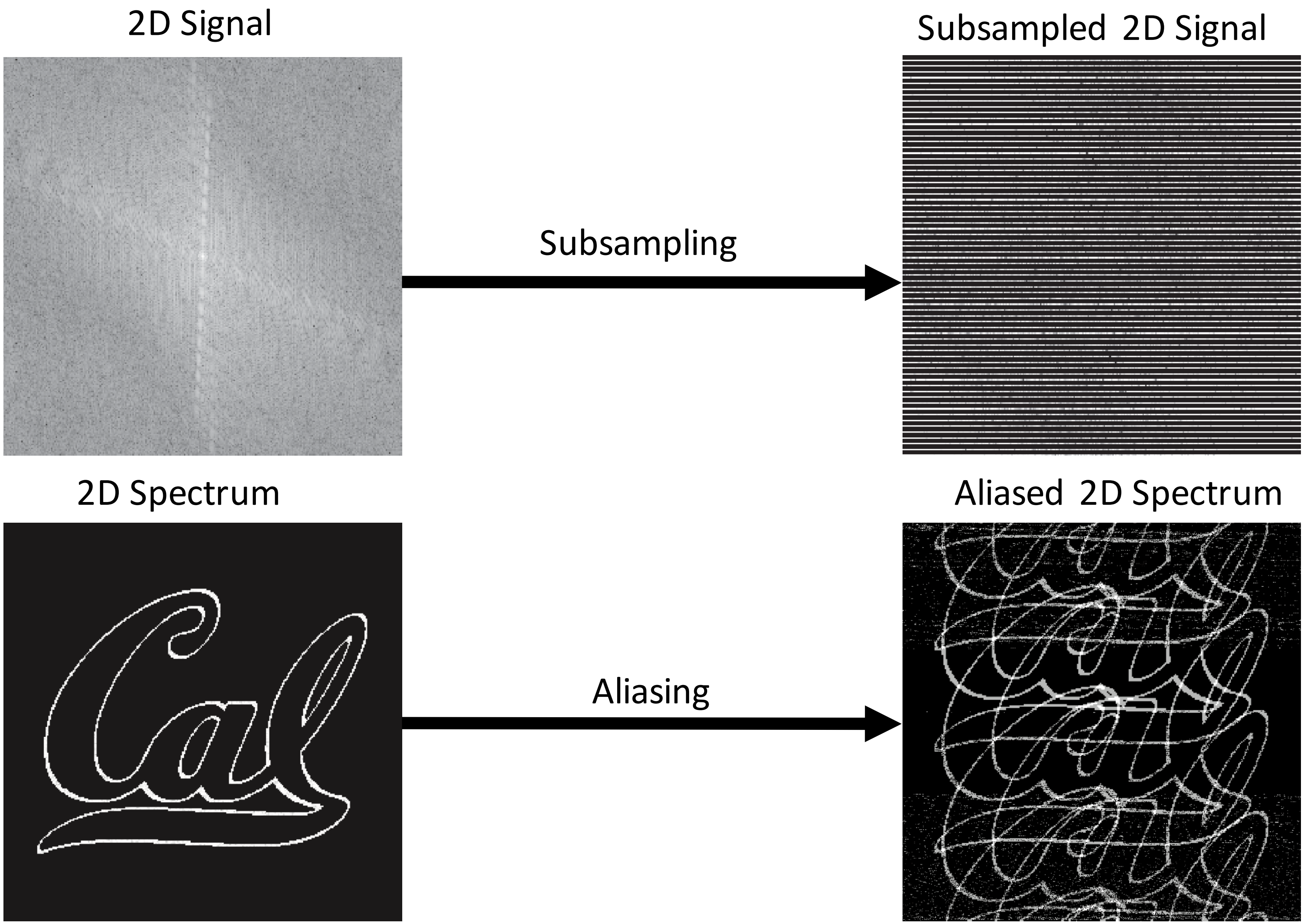
2D Signal



2D Spectrum

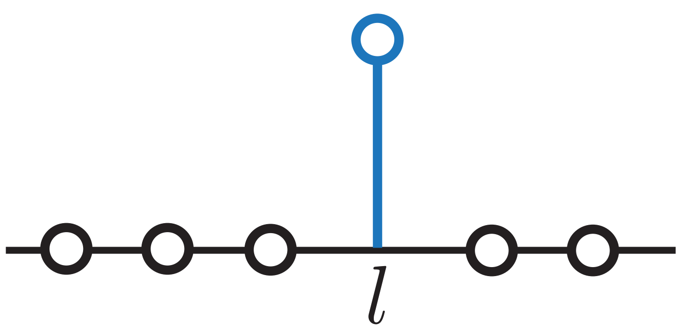
Aliasing with Sparse Spectrum

- Most entries do not have aliasing!



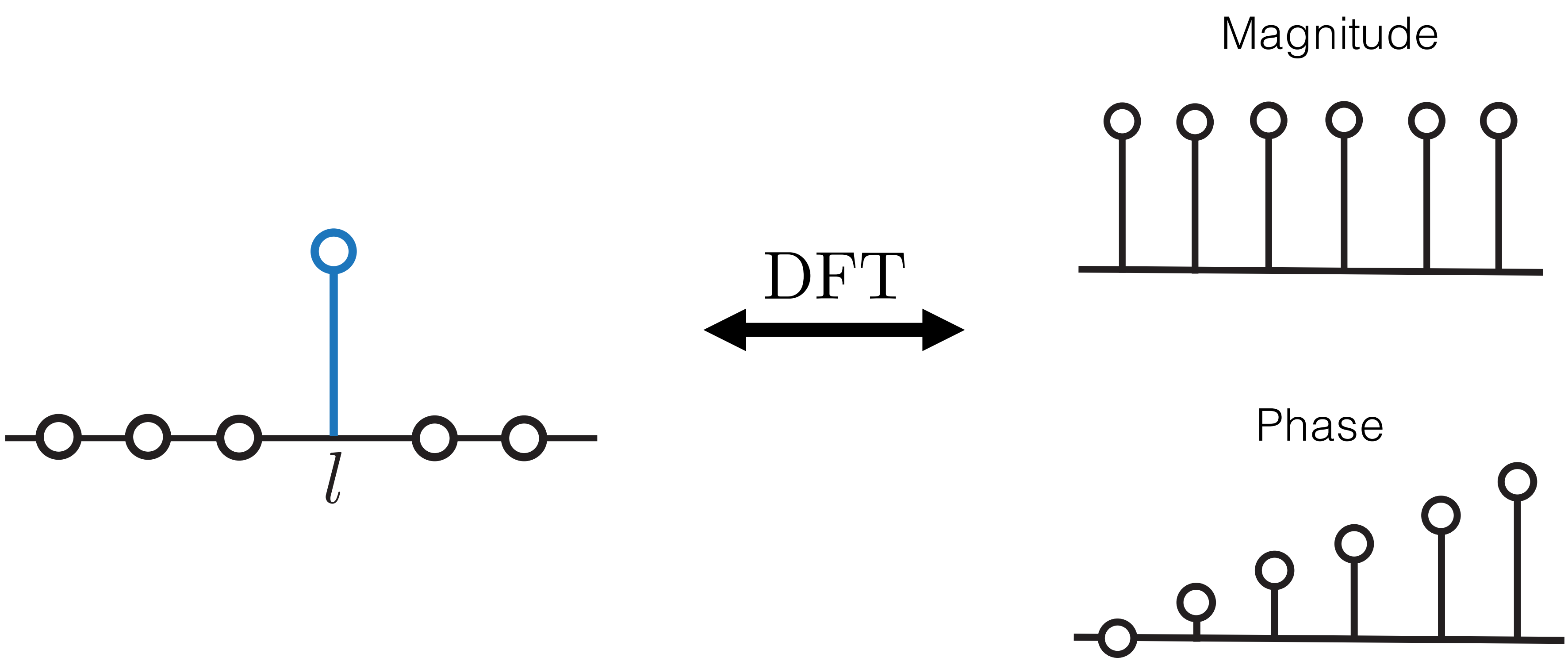
1-Sparse DFT

$$X[k] = \delta[k - l]$$



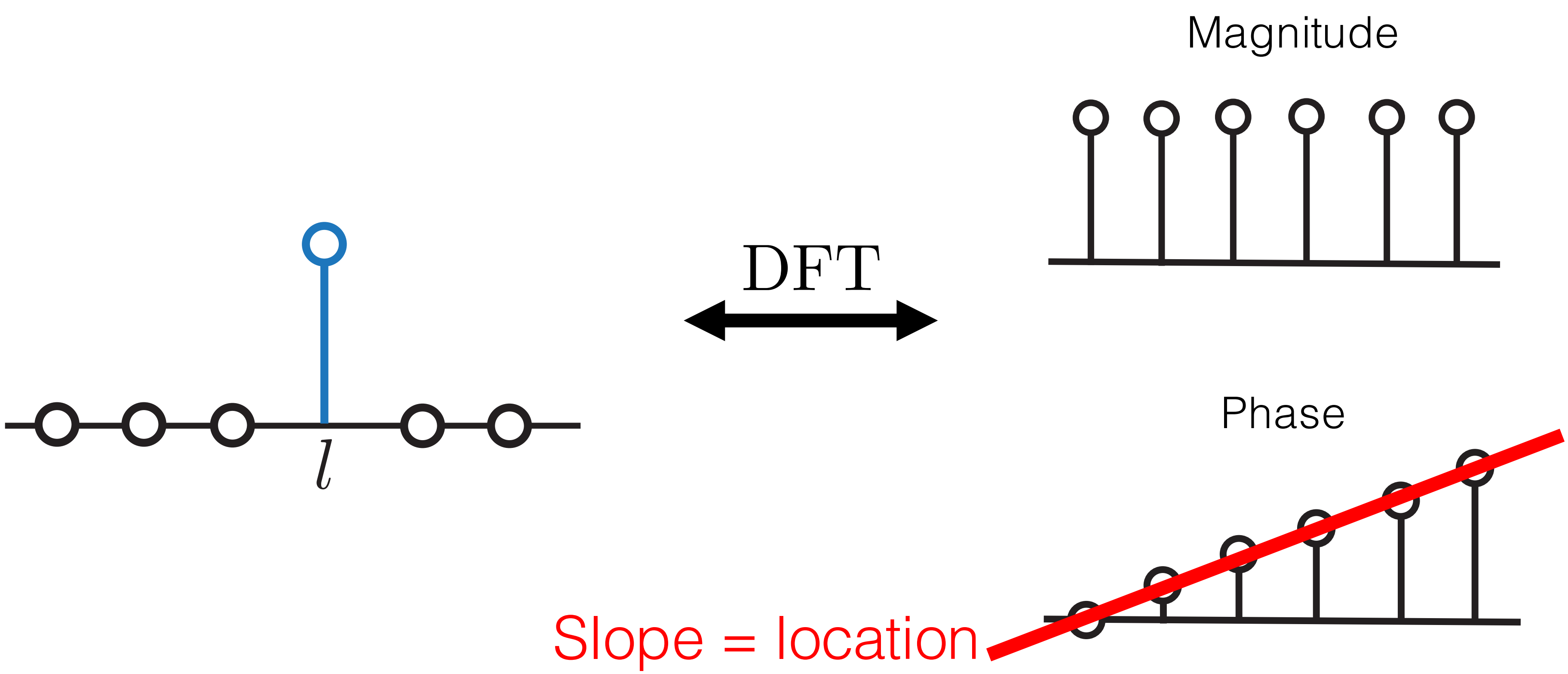
1-Sparse DFT

$$X[k] = \delta[k - l] \qquad x[n] = e^{j\frac{2\pi}{N}ln}$$



1-Sparse DFT

$$X[k] = \delta[k-l] \qquad\qquad x[n] = e^{j\frac{2\pi}{N}ln}$$

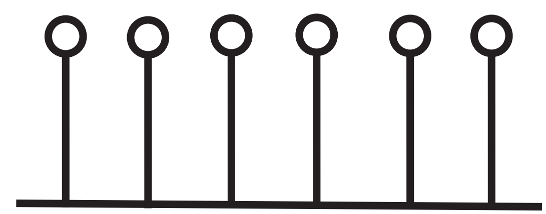


1-Sparse DFT

$$X[k] = \delta[k - l]$$

$$x[n] = e^{j\frac{2\pi}{N}ln}$$

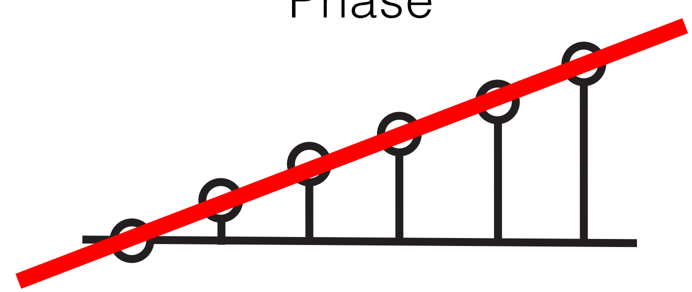
Magnitude



- For noise-less, needs only 2 samples

$$\angle(x[1]x^*[0]) = \frac{2\pi}{N}l$$

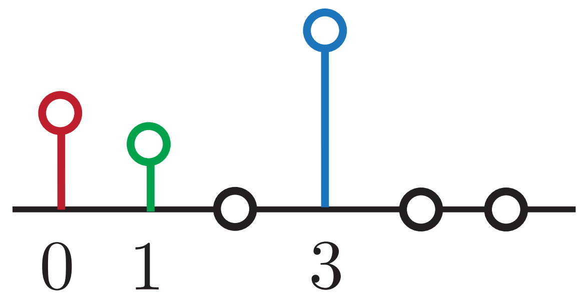
Phase



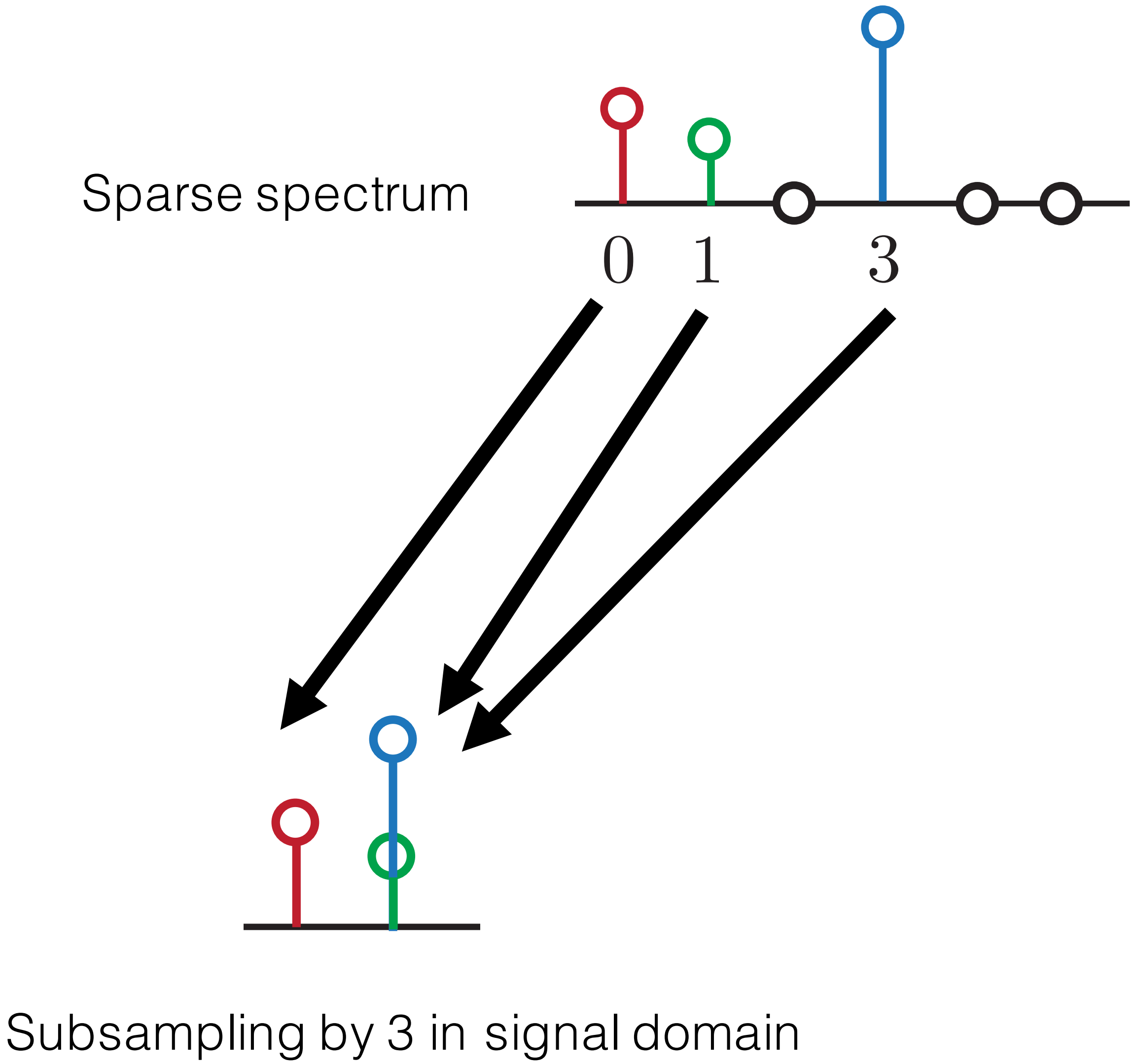
- Constant computation time

Different subsampling produces different aliasing

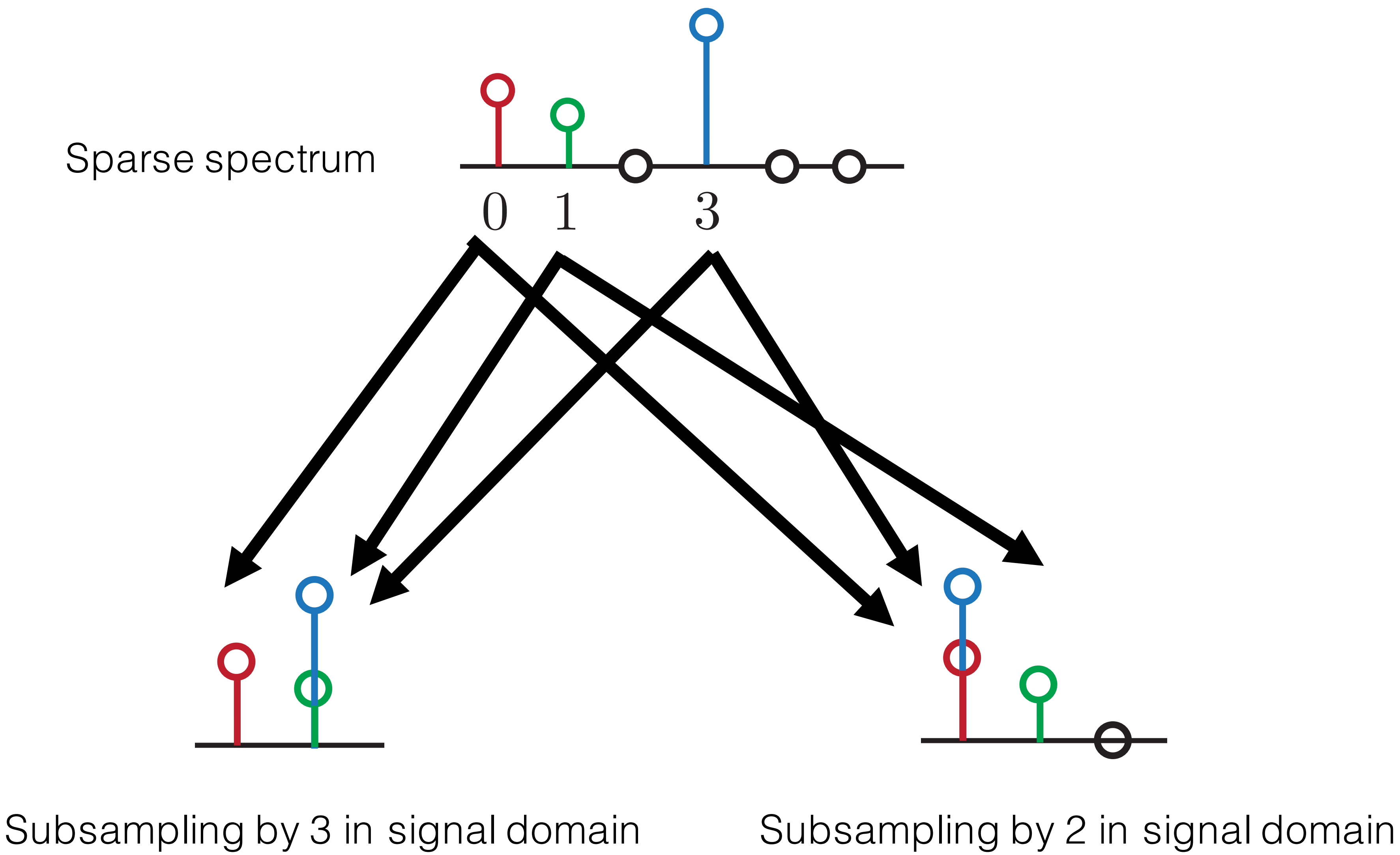
Sparse spectrum



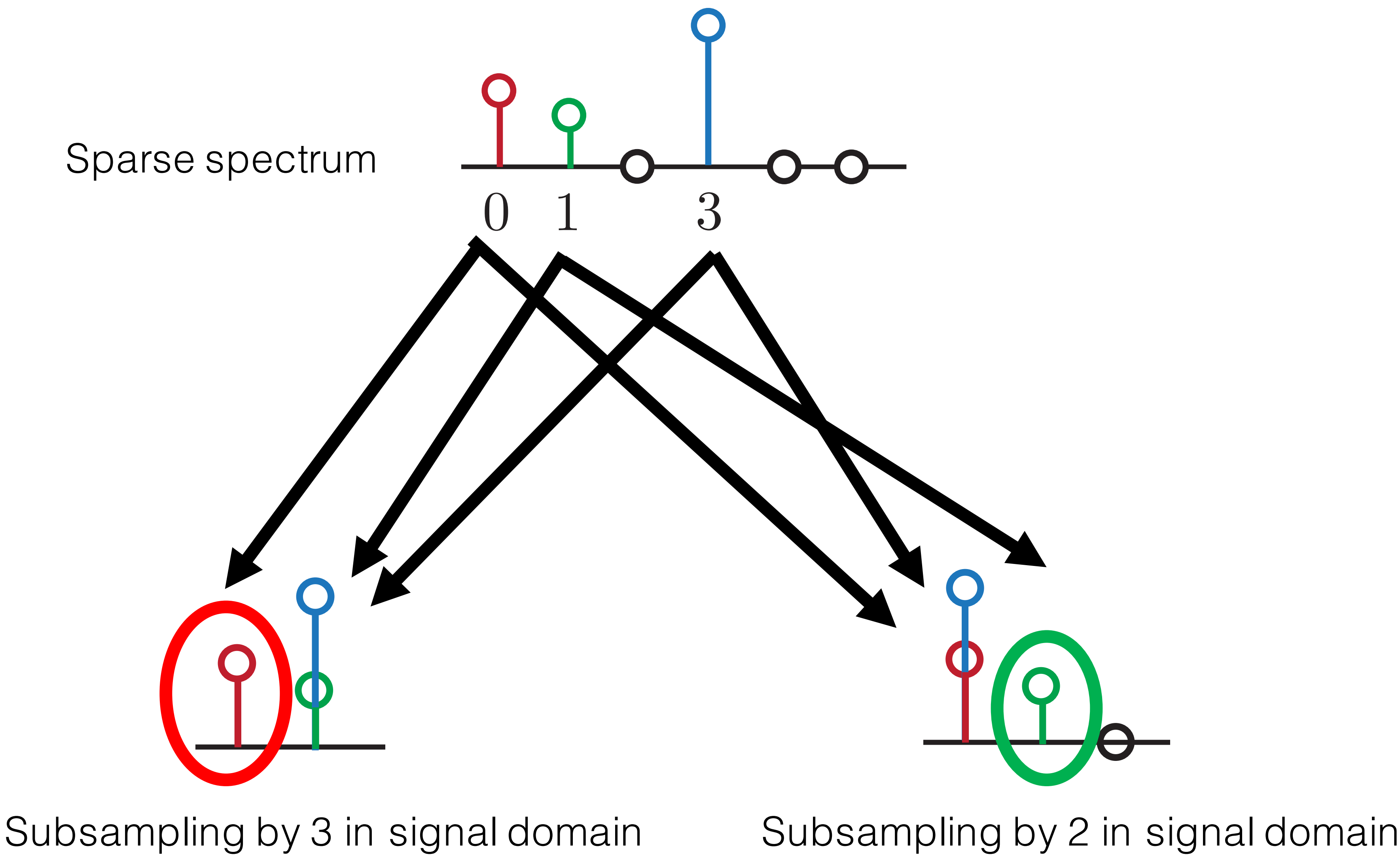
Different subsampling produces different aliasing



Different subsampling produces different aliasing



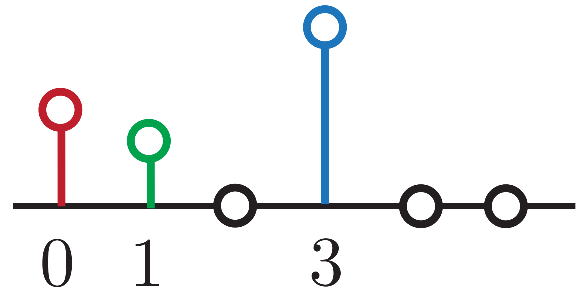
Different subsampling produces different aliasing



- Red and green are exposed with different subsampling

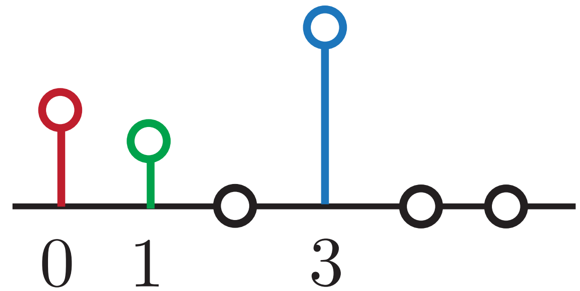
Combining three ideas

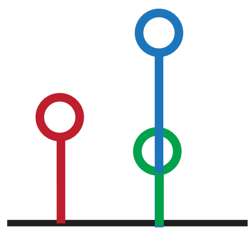
Sparse spectrum



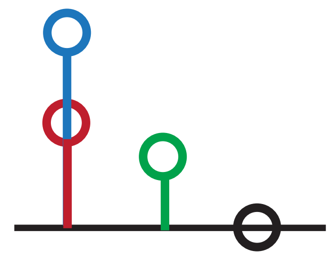
Combining three ideas

Sparse spectrum





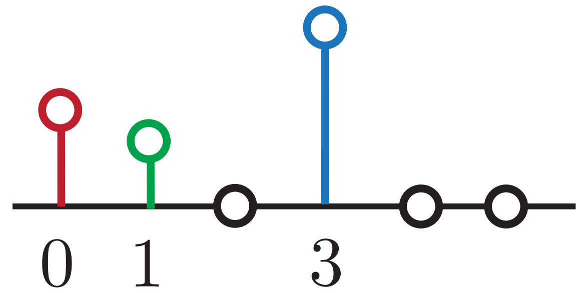
Subsampling by 3 in signal domain

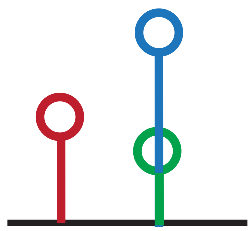


Subsampling by 2 in signal domain

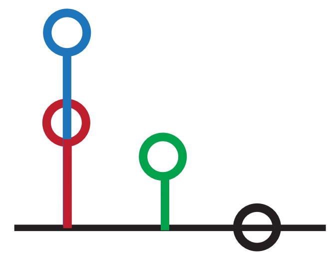
Combining three ideas

Sparse spectrum

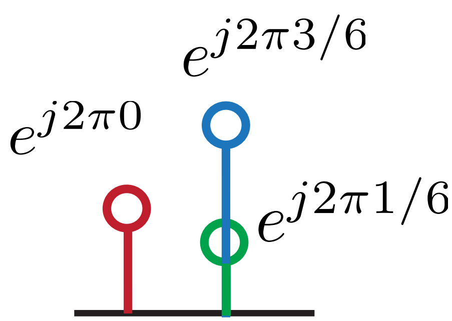




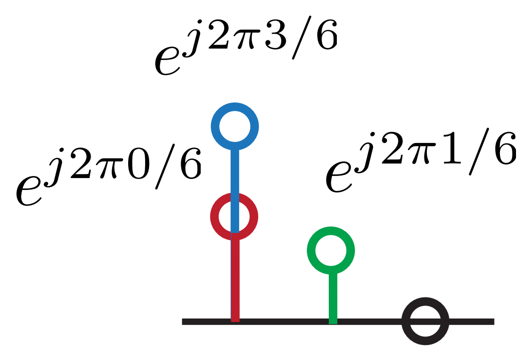
Subsampling by 3 in signal domain



Subsampling by 2 in signal domain



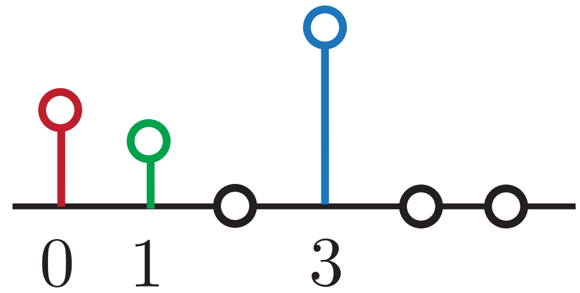
Shift sampling pattern by 1



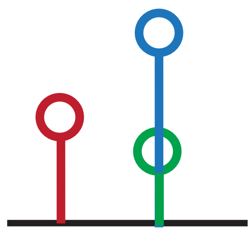
Shift sampling pattern by 1

Combining three ideas

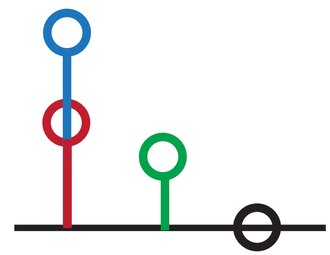
Sparse spectrum



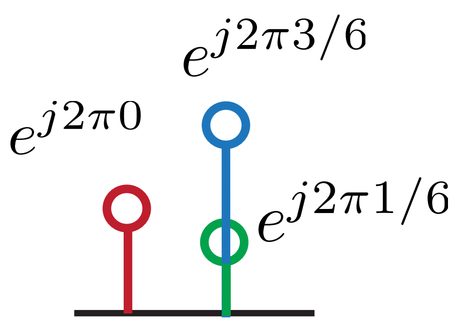
Can recover red and green locations via phase differences



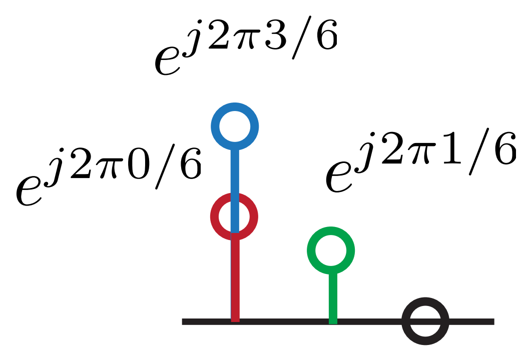
Subsampling by 3 in signal domain



Subsampling by 2 in signal domain



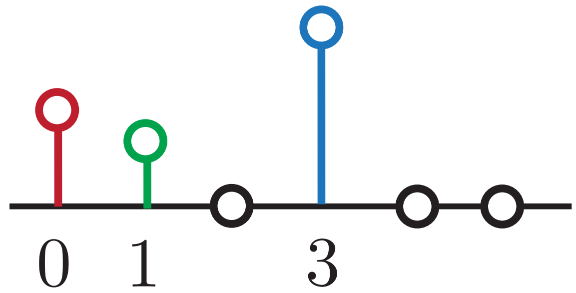
Shift sampling pattern by 1



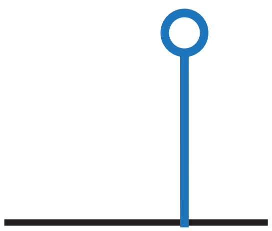
Shift sampling pattern by 1

Combining three ideas

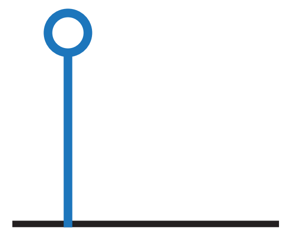
Sparse spectrum



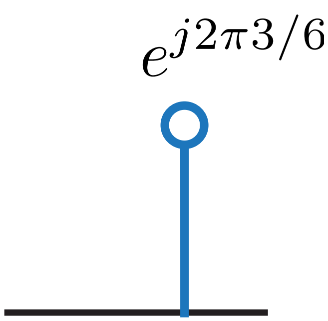
Can peel off red and green



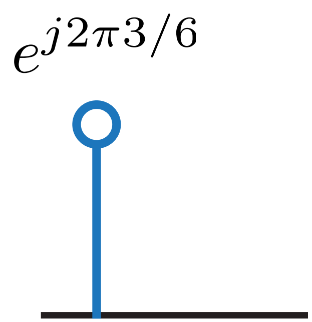
Subsampling by 3 in signal domain



Subsampling by 2 in signal domain



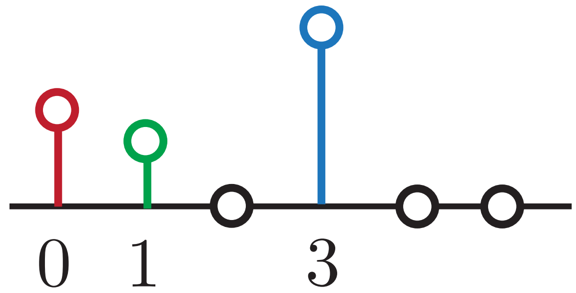
Shift sampling pattern by 1



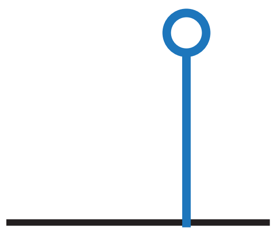
Shift sampling pattern by 1

Combining three ideas

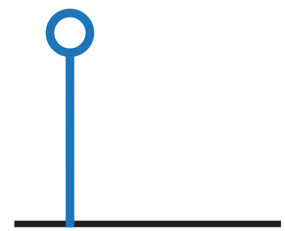
Sparse spectrum



Can recover blue location via phase differences

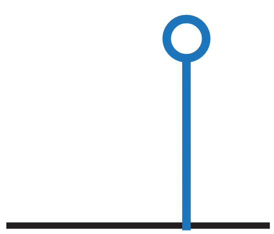


Subsampling by 3 in signal domain



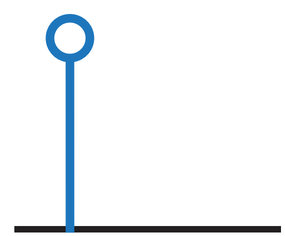
Subsampling by 2 in signal domain

$e^{j2\pi 3/6}$



Shift sampling pattern by 1

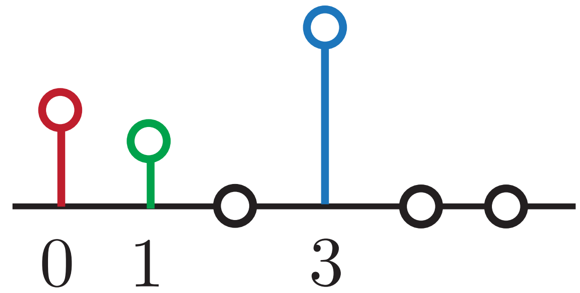
$e^{j2\pi 3/6}$



Shift sampling pattern by 1

Combining three ideas

Sparse spectrum



Recovered all sparse entries

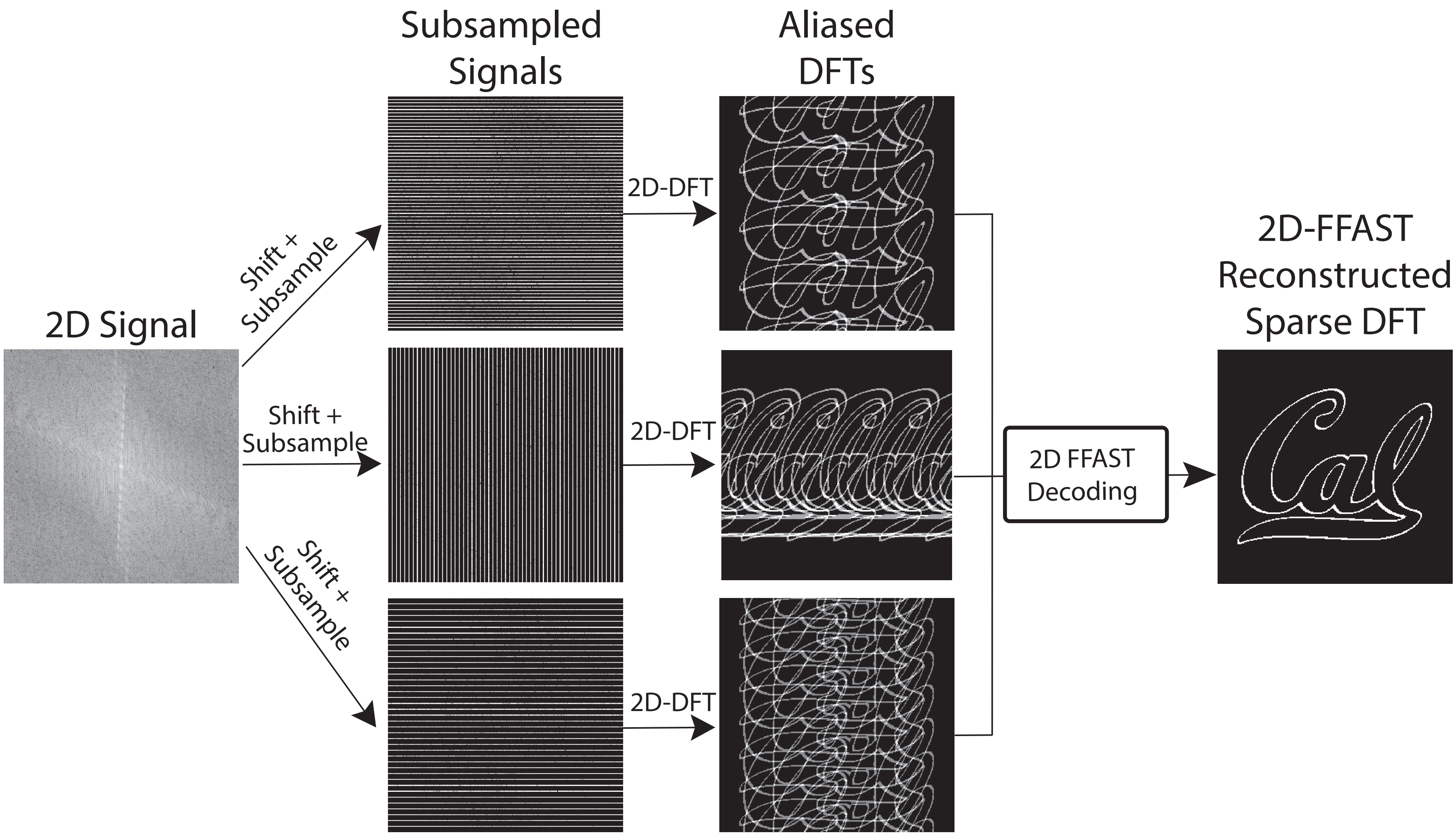
Subsampling by 3 in signal domain

Subsampling by 2 in signal domain

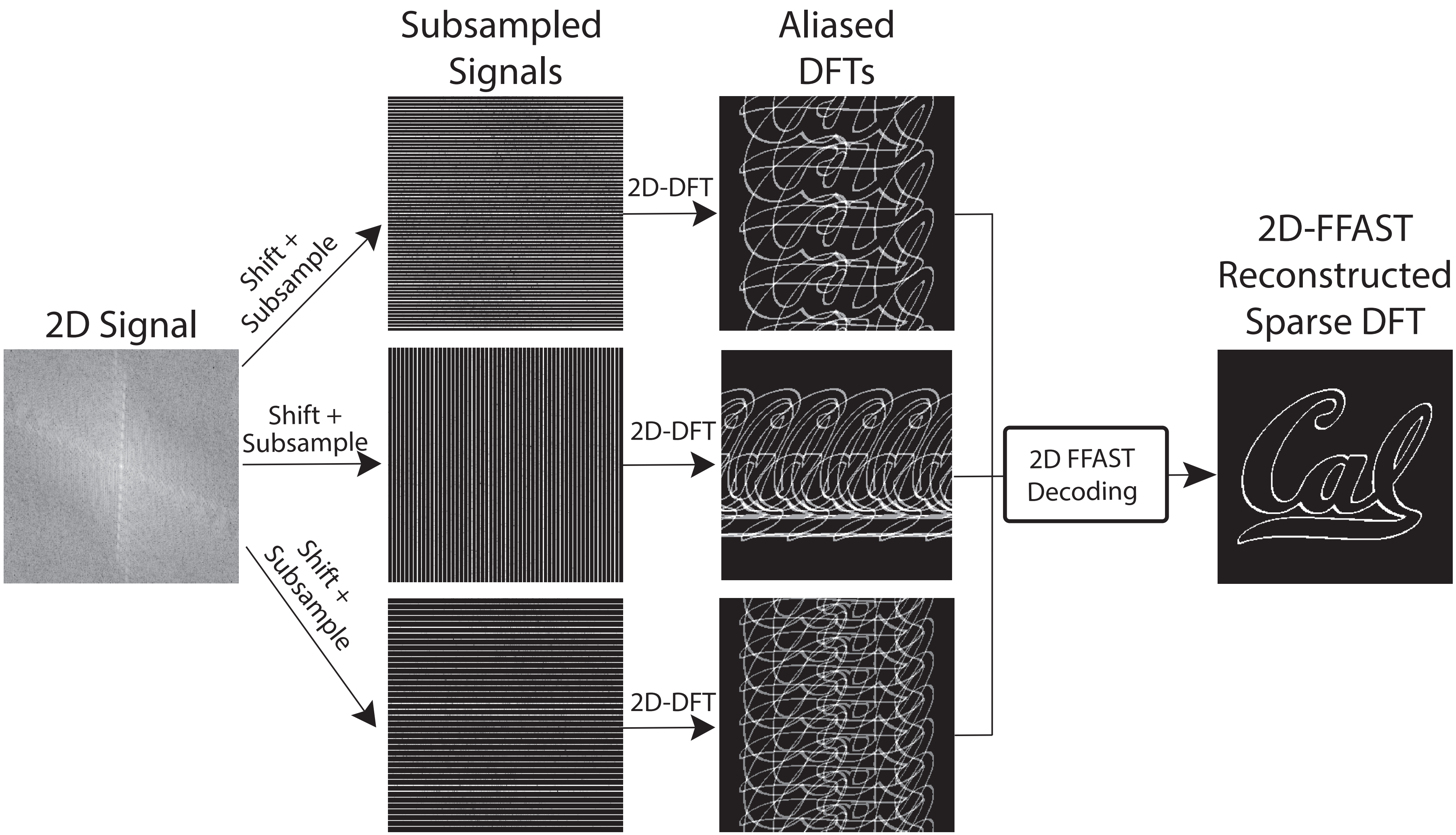
Shift sampling pattern by 1

Shift sampling pattern by 1

2D-FFAST Architecture



2D-FFAST Architecture



The more we sub-sample, the less computation we have!

Sampling Factors

- How to pick sampling factors to get diverse aliasing patterns?
- Clearly subsampling by 2 and 4 do not work

- 1D (Pawar and Ramchandran 2013):
  - Based on the Chinese Remainder Theorem
  - Sampling factors should be relatively co-prime
  - For example, subsample by 40, 41, and 43

- 2D (This work):
  - Product of 2D subsampling factors should be relatively co-prime
  - For example, subsample by $\underbrace{(5, 5)}_{25}$, $\underbrace{(6, 6)}_{36}$ and $\underbrace{(7, 7)}_{49}$

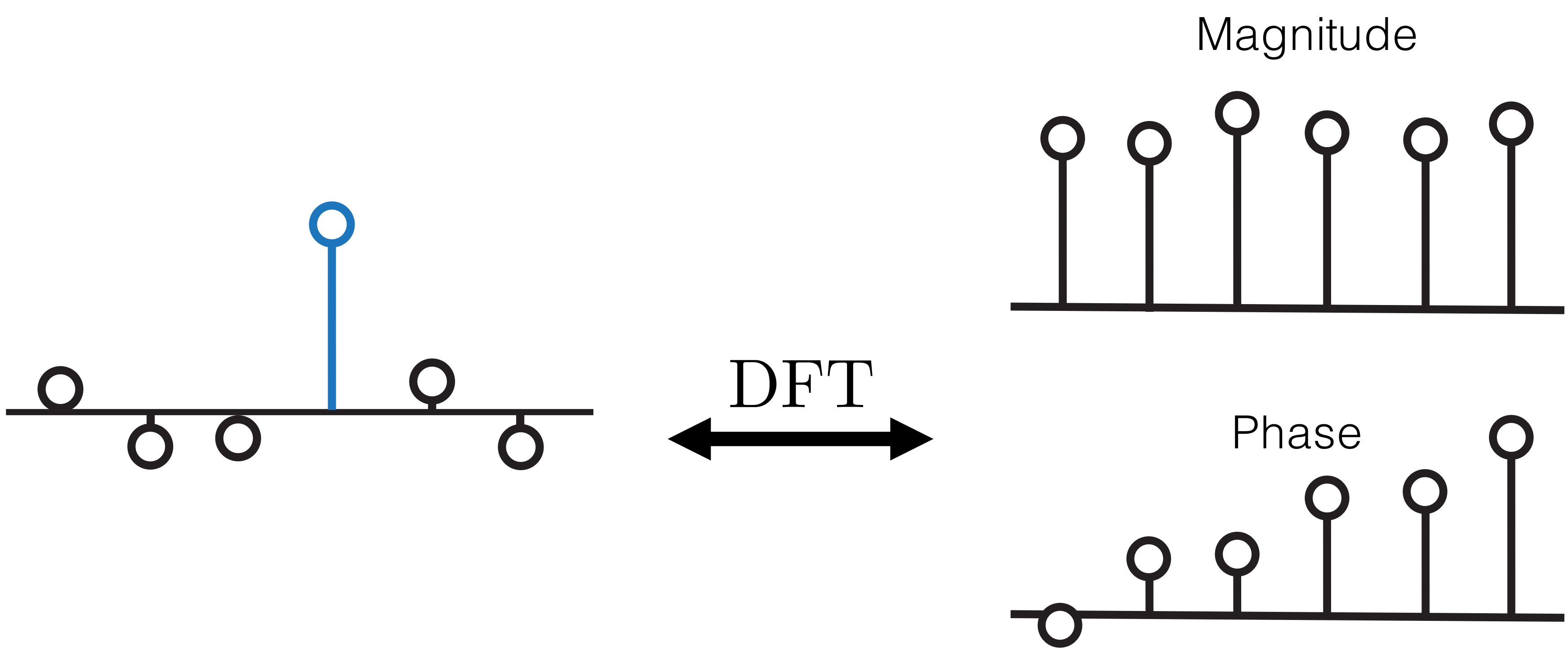
Theoretical Guarantee

- For a input that satisfies the dimension assumption
- The 2D-FFAST computes its k-sparse 2D-DFT w.h.p:
    - ~4k measurements for almost all sublinear sparsity
    - O(k log k) computation complexity
- For more details: http://arxiv.org/abs/1509.05849v1.

Noisy 1-sparse DFT

$X[k] = \delta[k-l]$ + noise

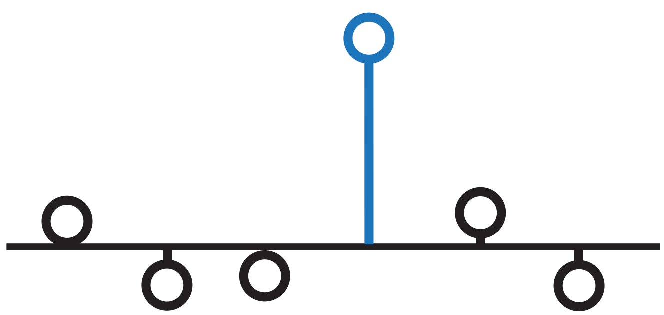
$x[n] = e^{j\frac{2\pi}{N}ln}$ + noise



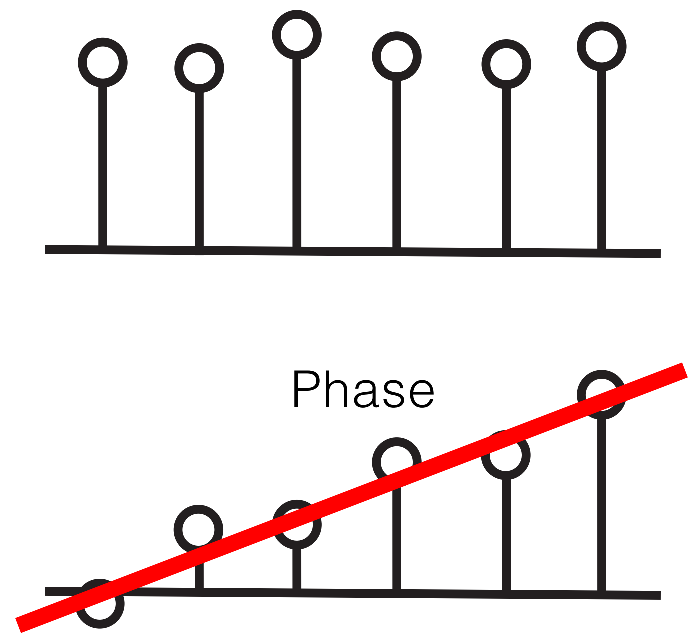
Noisy 1-sparse DFT

$X[k] = \delta[k-l]$ + noise

$x[n] = e^{j\frac{2\pi}{N}ln}$ + noise



$\xleftrightarrow{\text{DFT}}$



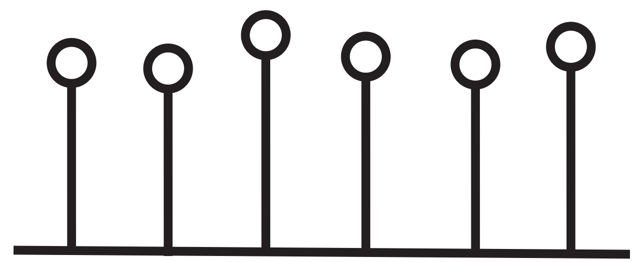
Noisy 1-sparse DFT

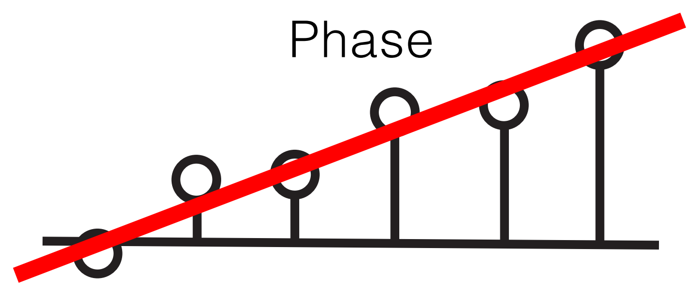
More samples to get robust location estimation

$$X[k] = \delta[k-l] \text{ + noise}$$

$$x[n] = e^{j\frac{2\pi}{N}ln} \text{ + noise}$$

Magnitude





- To recover the location,

$$\angle(x[1]x^*[0]) = \frac{2\pi}{N}l \text{ + noise}$$

$$\angle(x[2]x^*[1]) = \frac{2\pi}{N}l \text{ + noise}$$

$$\angle(x[3]x^*[2]) = \frac{2\pi}{N}l \text{ + noise}$$

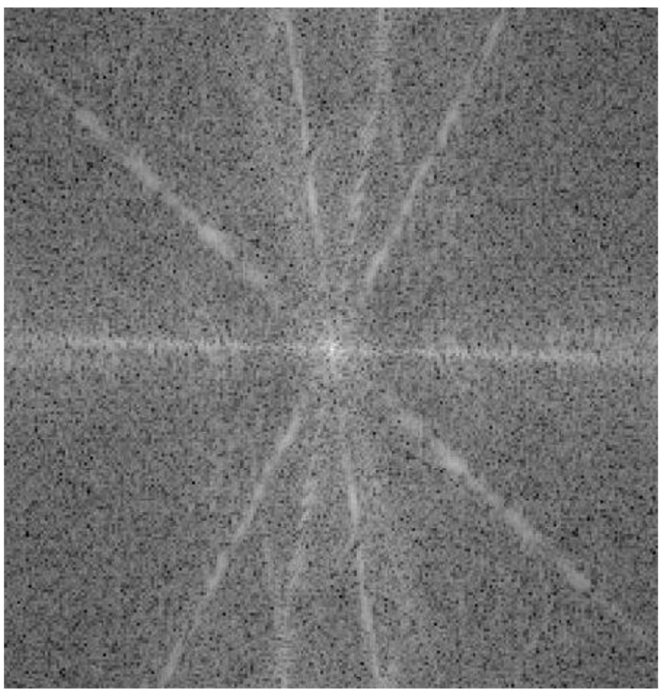
Theoretical Guarantee

- For an N = Nx x Ny input
- The 2D-FFAST computes its k-sparse 2D-DFT w.h.p:
    - $O(k \log^3 N)$ measurements
    - $O(k \log^4 N)$ computation complexity
- For more details: http://arxiv.org/abs/1509.05849v1.

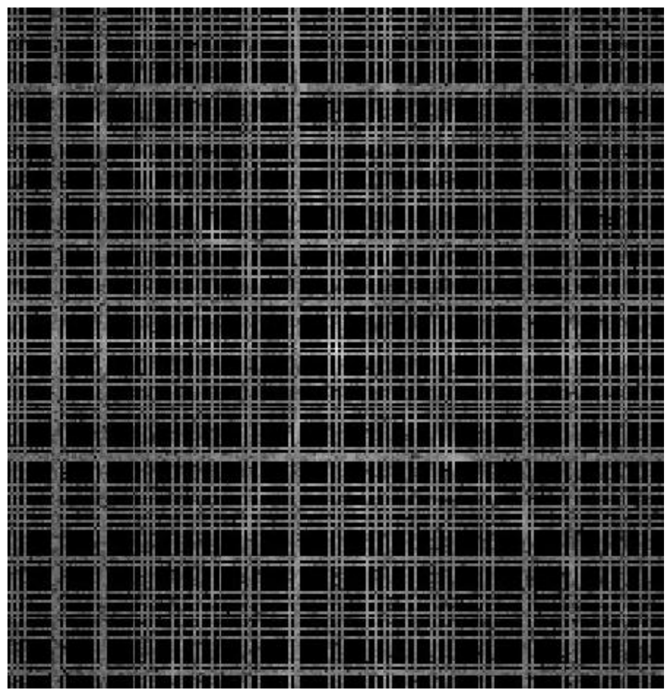
Simulation Results

Image size:	247 x 238 = 58,786
Sparsity:	4599
Measurements:	5.46 k = 25,126

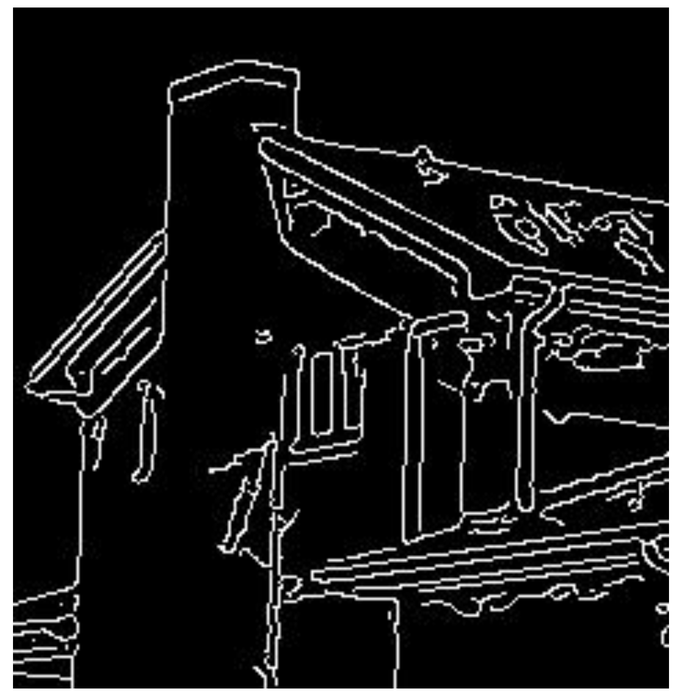
Original Signal



Sub-sampled Signal



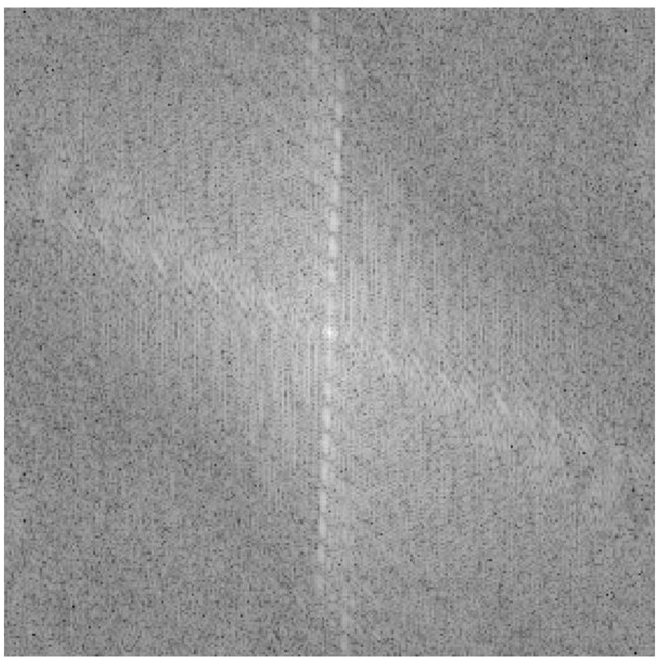
Reconstructed Spectrum



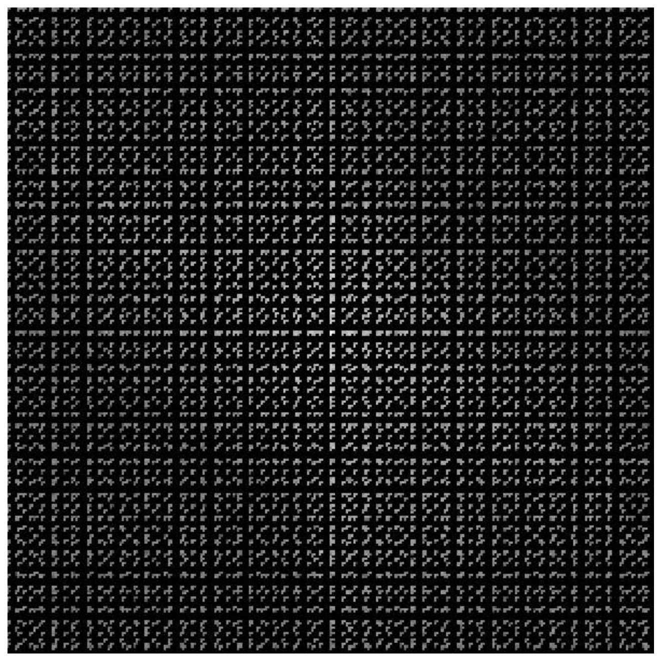
Simulation Results

Image size:	280 x 280 = 78,400
Sparsity:	3509
Measurements:	4.75 k = 16,668

Original Signal

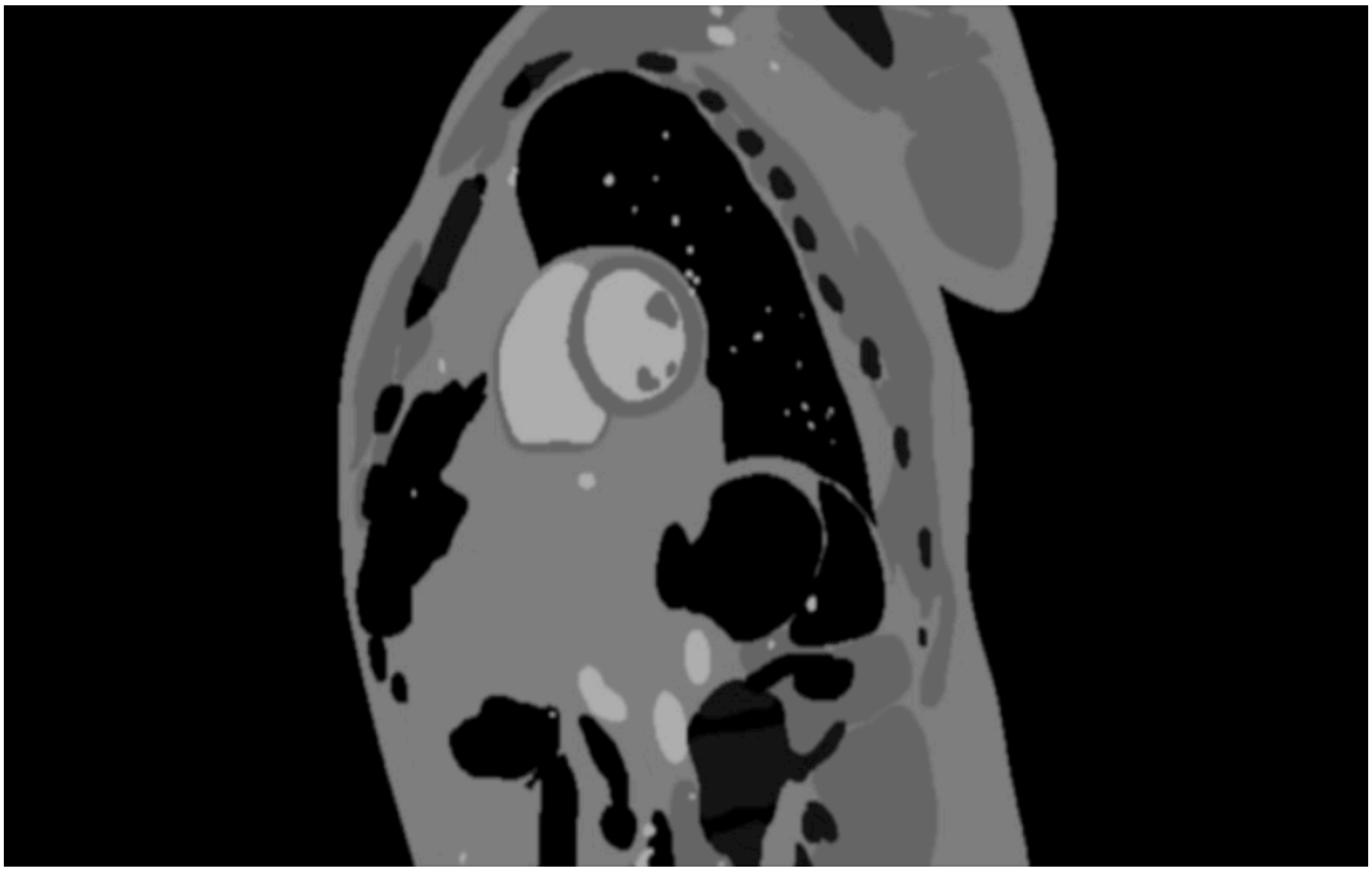


Sub-sampled Signal



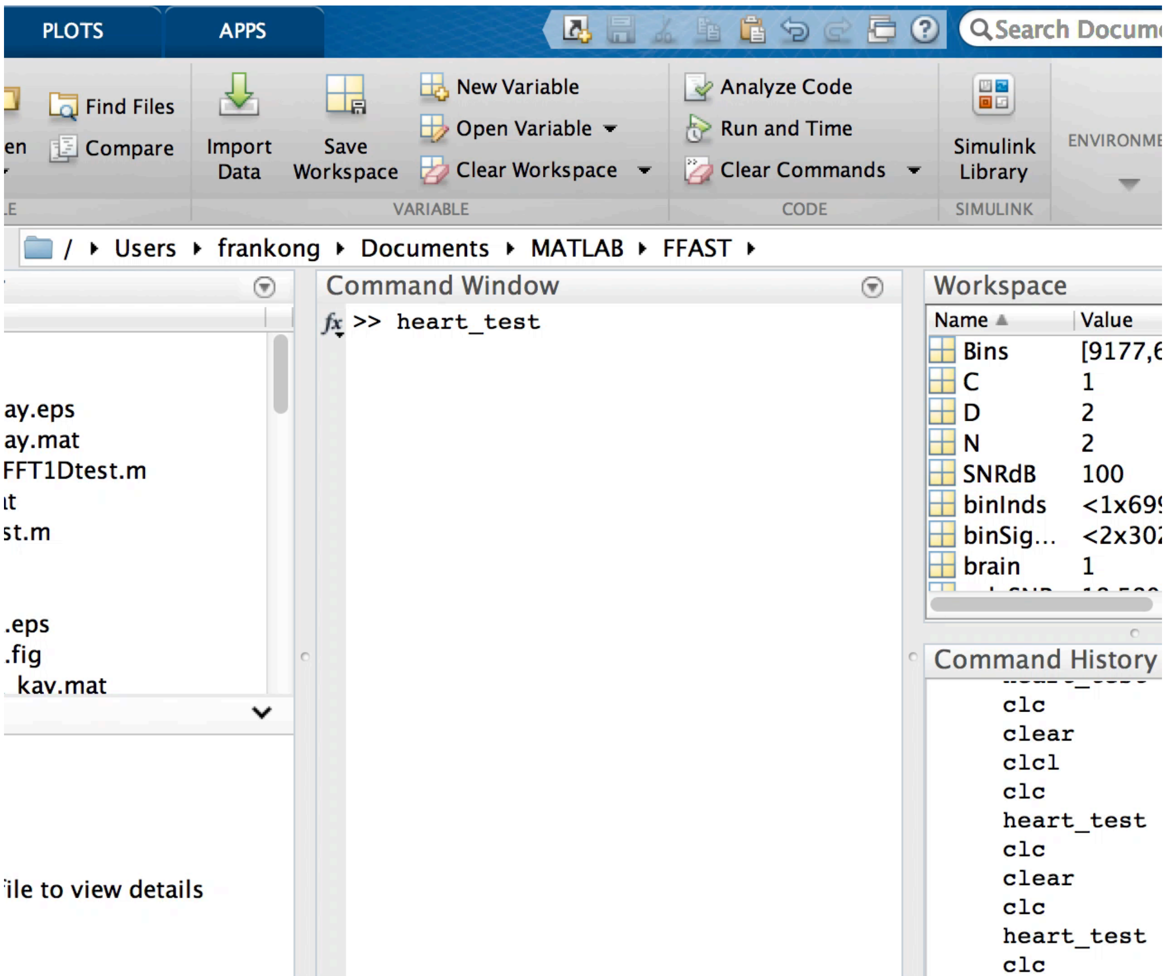
Reconstructed Spectrum

Simulation Results



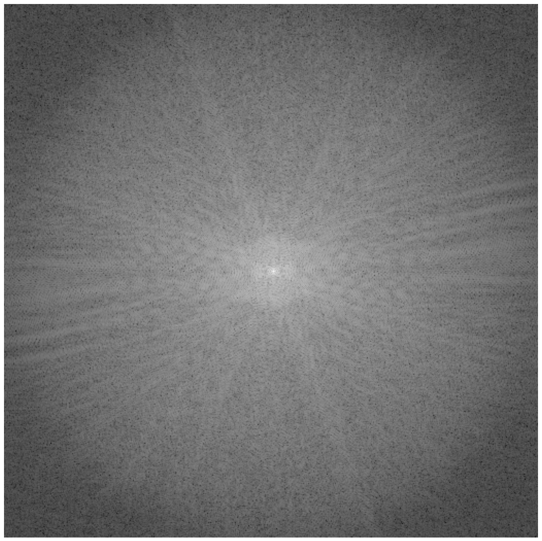
Simulation Results

Simulation Results

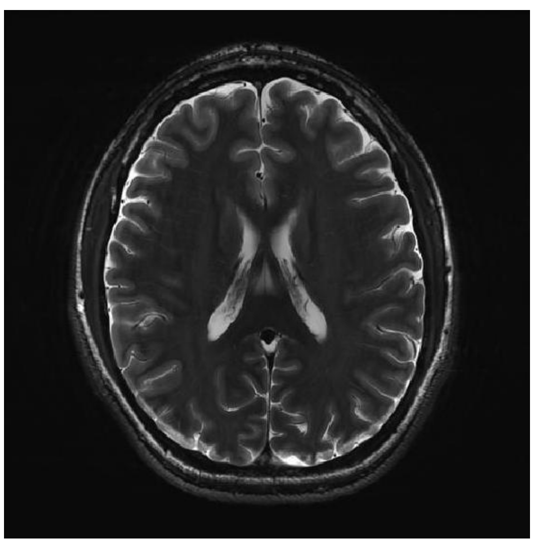


Simulation Results Beyond Signal Model

Original Signal

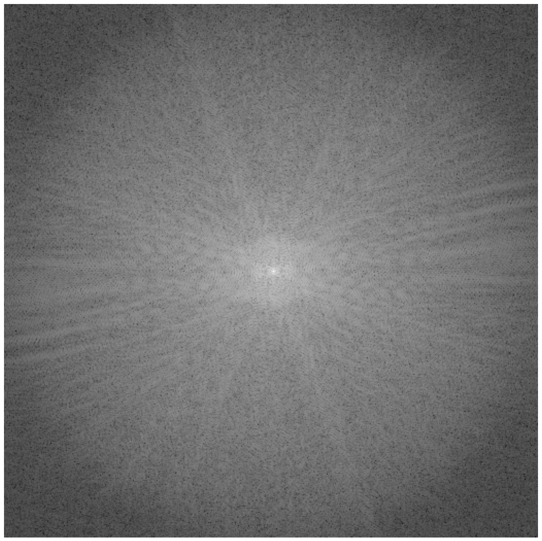


Original Spectrum

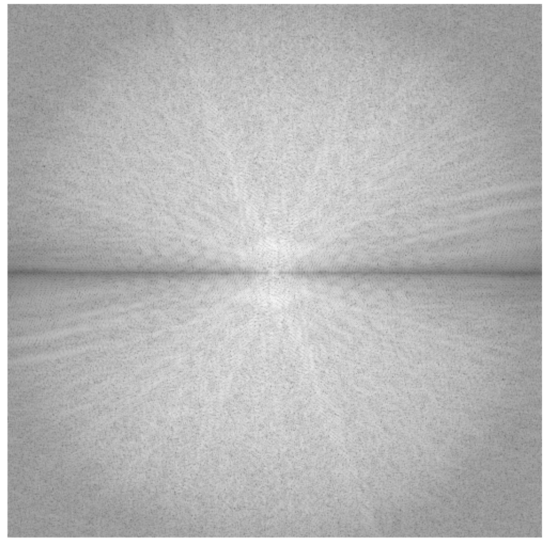


Simulation Results Beyond Signal Model

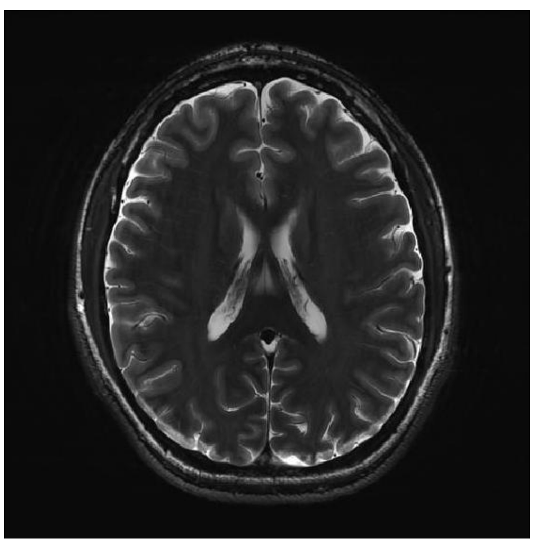
Original Signal



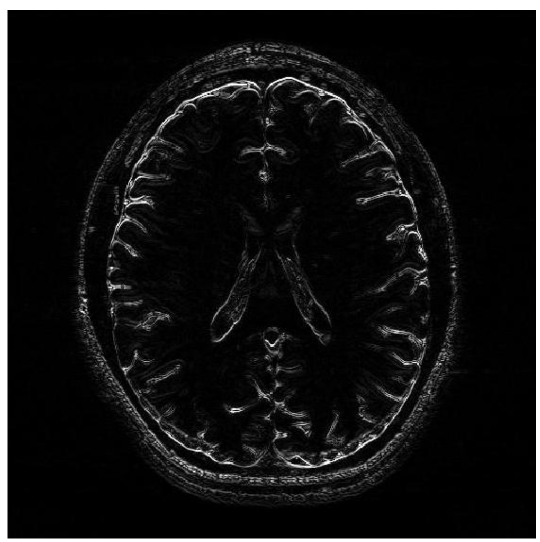
Difference filtered Signal



Original Spectrum

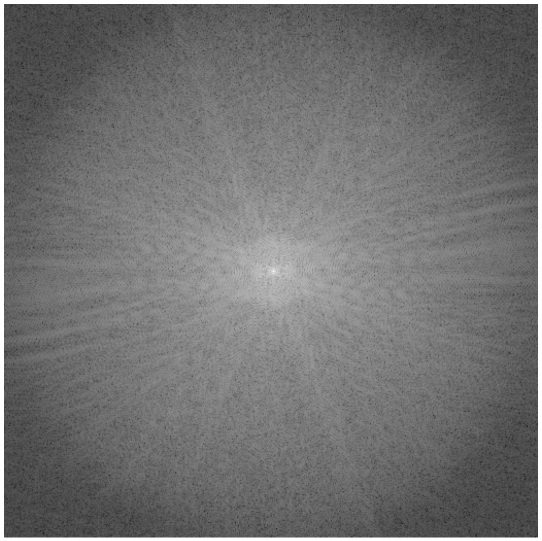


Difference filtered Spectrum

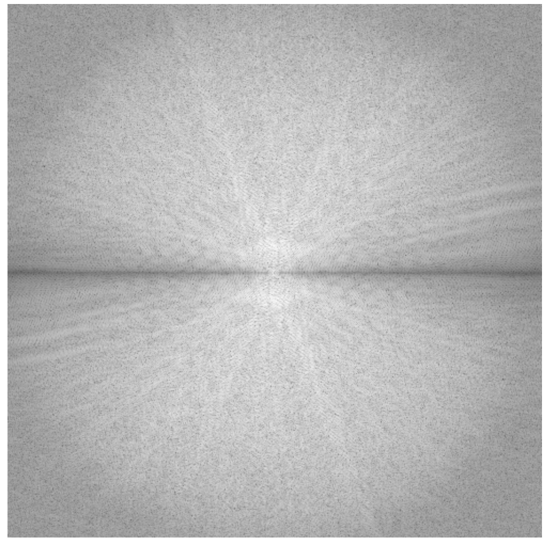


Simulation Results Beyond Signal Model

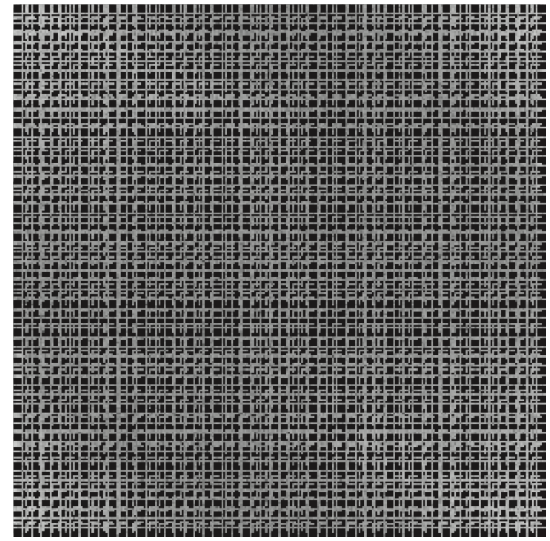
Original Signal



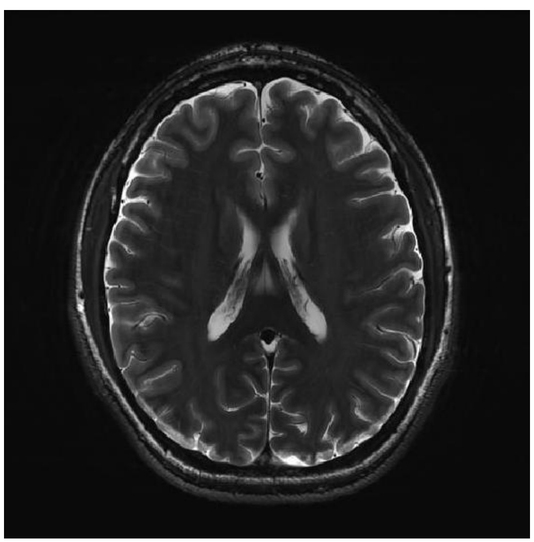
Difference filtered Signal



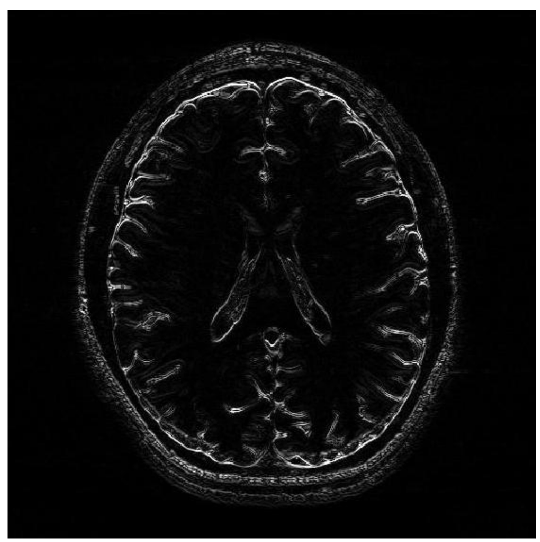
FFAST subsampled Signal



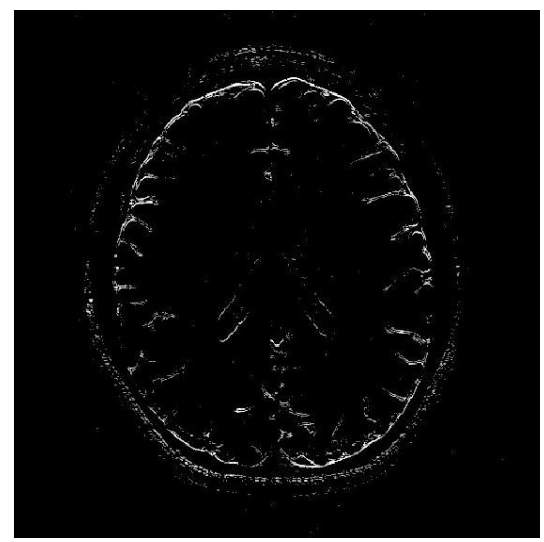
Original Spectrum



Difference filtered Spectrum

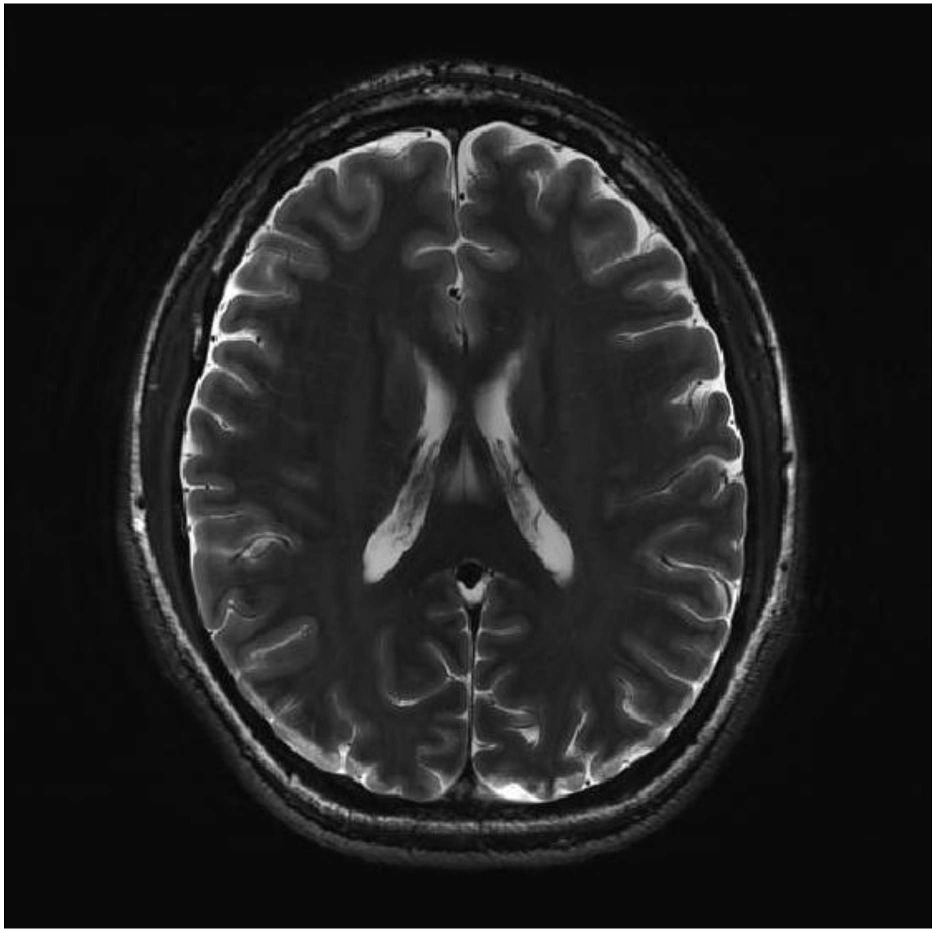


FFAST reconstructed spectrum

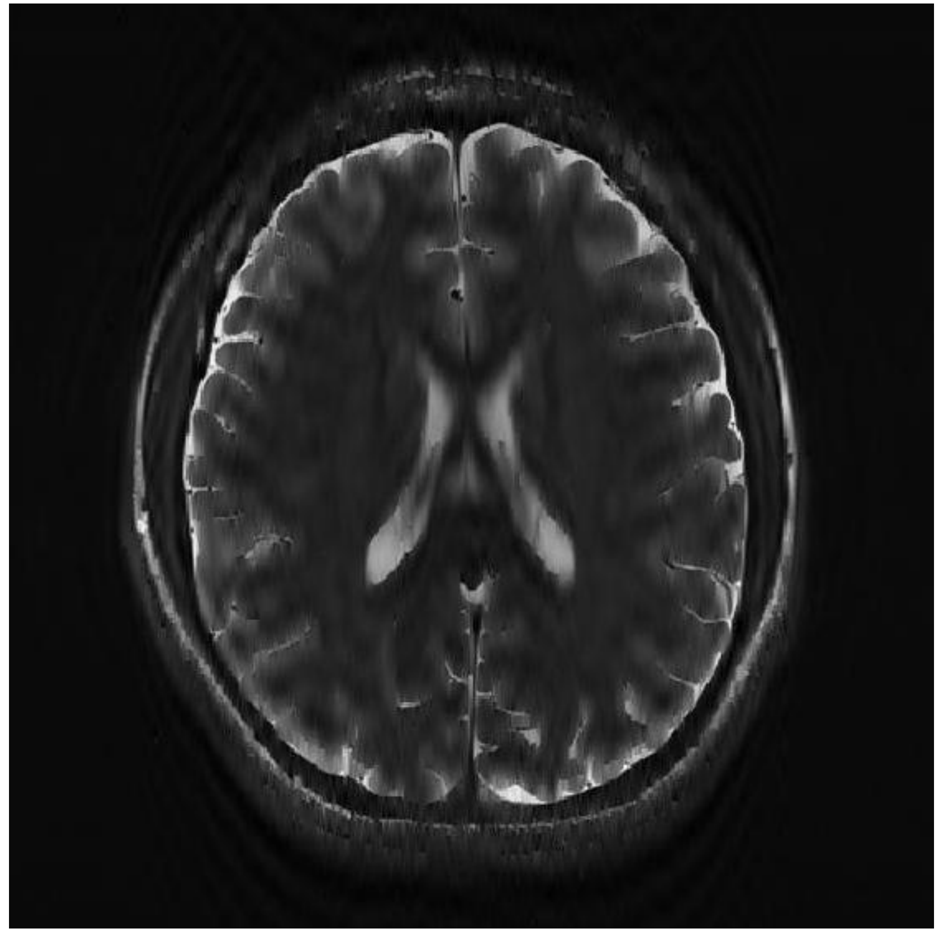


Simulation Results Beyond Signal Model

Original Spectrum



FFAST reconstructed spectrum



Not the best quality you can get with CS but FFAST does it fast!

Promising initial result!

Conclusion

Thank you!

- Fast in both acquisition and reconstruction
- Illustrate 2D-FFAST architecture through 3 key ideas
- Coding theory guided reconstruction method
- https://github.com/UCBASiCS/FFAST.git