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

Frank Ong, Sameer Pawar and Kannan Ramchandran
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!

Loading...
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

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

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

![Diagram of DFT with magnitude and phase components]
1-Sparse DFT

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

\[ x[n] = e^{j \frac{2\pi}{N} ln} \]
1-Sparse DFT

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

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

- For noise-less, needs only 2 samples

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

- Constant computation time
Different subsampling produces different aliasing

Sparse spectrum

0  1  3
Different subsampling produces different aliasing

Sparse spectrum

Subsampling by 3 in signal domain
Different subsampling produces different aliasing

Sparse spectrum

Subsampling by 3 in signal domain

Subsampling by 2 in signal domain
Different subsampling produces different aliasing

Sparse spectrum

0 1 3

Subsampling by 3 in signal domain
Subsampling by 2 in signal domain

- Red and green are exposed with different subsampling
Combining three ideas

Sparse spectrum

0 1 3
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

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

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

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

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

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

Shift sampling pattern by 1

Shift sampling pattern by 1
Combining three ideas

Sparse spectrum

0 1 3

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 Signal

Shift + Subsample

Subsampled Signals

Aliased DFTs

2D-DFT

2D-FFAST Reconstructed Sparse DFT

Cal
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 (5, 5), (6, 6) and (7, 7)

  25  36  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

Noisy 1-sparse DFT

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

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

Magnitude

Phase
Noisy 1-sparse DFT

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

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

Magnitude

Phase

Slope \approx \text{location}
Noisy 1-sparse DFT

More samples to get robust location estimation

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

- To recover the location,

\[ \angle(x[1]x^*[0]) = \frac{2\pi}{N} l + \text{noise} \]
\[ \angle(x[2]x^*[1]) = \frac{2\pi}{N} l + \text{noise} \]
\[ \angle(x[3]x^*[2]) = \frac{2\pi}{N} l + \text{noise} \]

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

Magnitude

Phase
Theoretical Guarantee

• For an $N = N_x \times N_y$ 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

Simulation Results

- Image size: $247 \times 238 = 58,786$
- Sparsity: 4599
- Measurements: $5.46 \, k = 25,126$

Images:
- Original Signal
- Sub-sampled Signal
- Reconstructed Spectrum
Simulation Results

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

• 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](https://github.com/UCBASiCS/FFAST.git)