Average-Case Sparse Fourier Transform Algorithms

Piotr Indyk

MIT

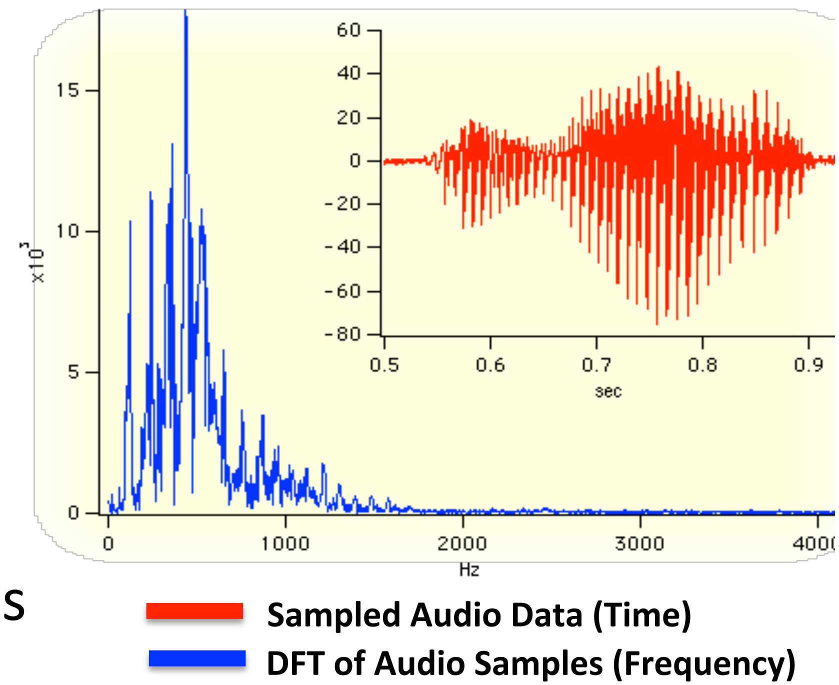
Ghazi, Hassanieh, Indyk, Katabi, Price, Lixin, "Sample-Optimal Average-Case Sparse Fourier Transform in 2D"

Fourier Transform

- Discrete Fourier Transform:
  - Given: a signal x[1…n]
  - Goal: compute the frequency vector x^ where

$$x\hat{}_f = \Sigma_t\, x_t\, e^{-2\pi i\, tf/n}$$

- Sparse Fourier Transform
  - Only "few" "large" coefficients
  - GL89, KM90, Mansour'92, GGIMS'02, AGS'03, GMS'05, Iwen'10, Akavia'10, HIKP'12,BCGLS'12,LWC'12…



SFFT

[Hassanieh, Indyk, Katabi, Price'12]

- All algorithms randomized, with constant probability of success, n is a power of 2
- Exactly k-sparse case : $O(k \log n)$

  (SFFT 3.0)
- Approximately k-sparse case
  - Let $Err^k_2(x\hat{}) = \min_{k\text{-sparse } z\hat{}} ||x\hat{} - z\hat{}||_2$
  - l_2/l_2 guarantee $||x\hat{}-y\hat{}||_2 \le C\, Err^k_2(x\hat{})$: $O(k \log(n) \log(n/k))$ time

    (SFFT 4.0)
  - Weaker results for l_∞/l_2 guarantee

    (SFFT 1.0 and SFFT 2.0)

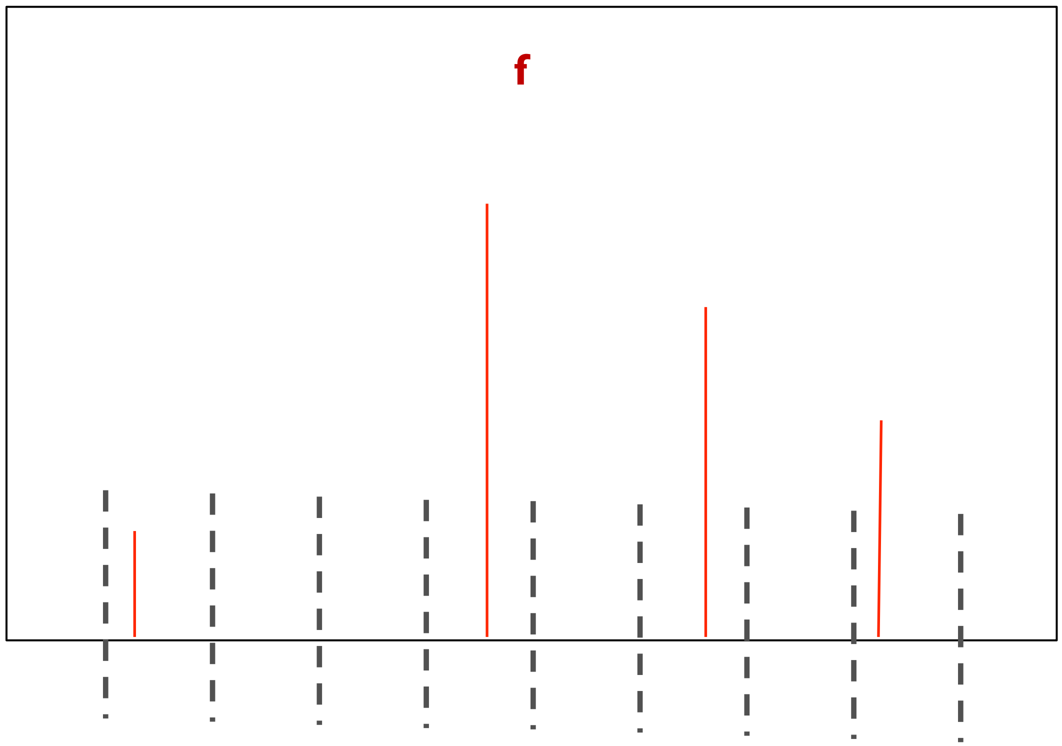
Sample complexity?

Algorithm	Time	Samples	Lower bound
SFFT 3.0 (exact)	O(k log n)	O(k log n)	k
SFFT 4.0 (robust)	O(k log(n) log(n/k))	O(k log(n) log(n/k))	k log (n/k)

How Does Sparse FFT Work?

1- Bucketize

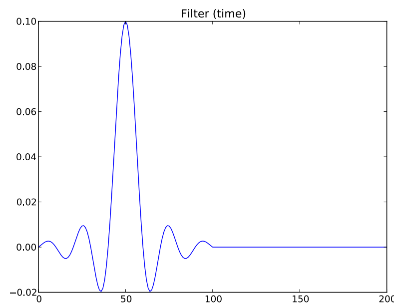
Divide spectrum into a few buckets

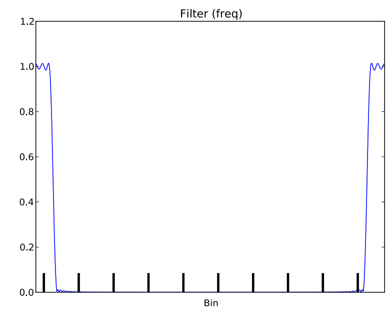


2- Estimate

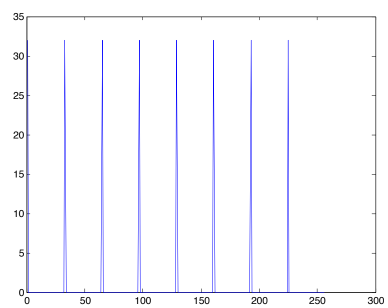
Estimate the large coefficient (position and value) in each non-empty bucket

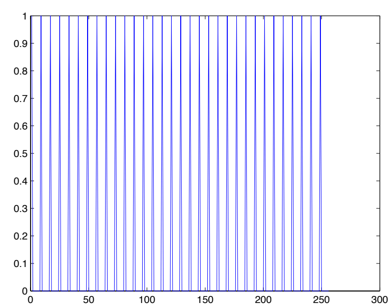
Bucketization





- For F = Sinc x Gaussian, $|\mathrm{supp}^{>1/n}(F)| * |\mathrm{supp}^{>1/n}(F\hat{})| = n\log n$
  - This leads to O(k log n) sample complexity

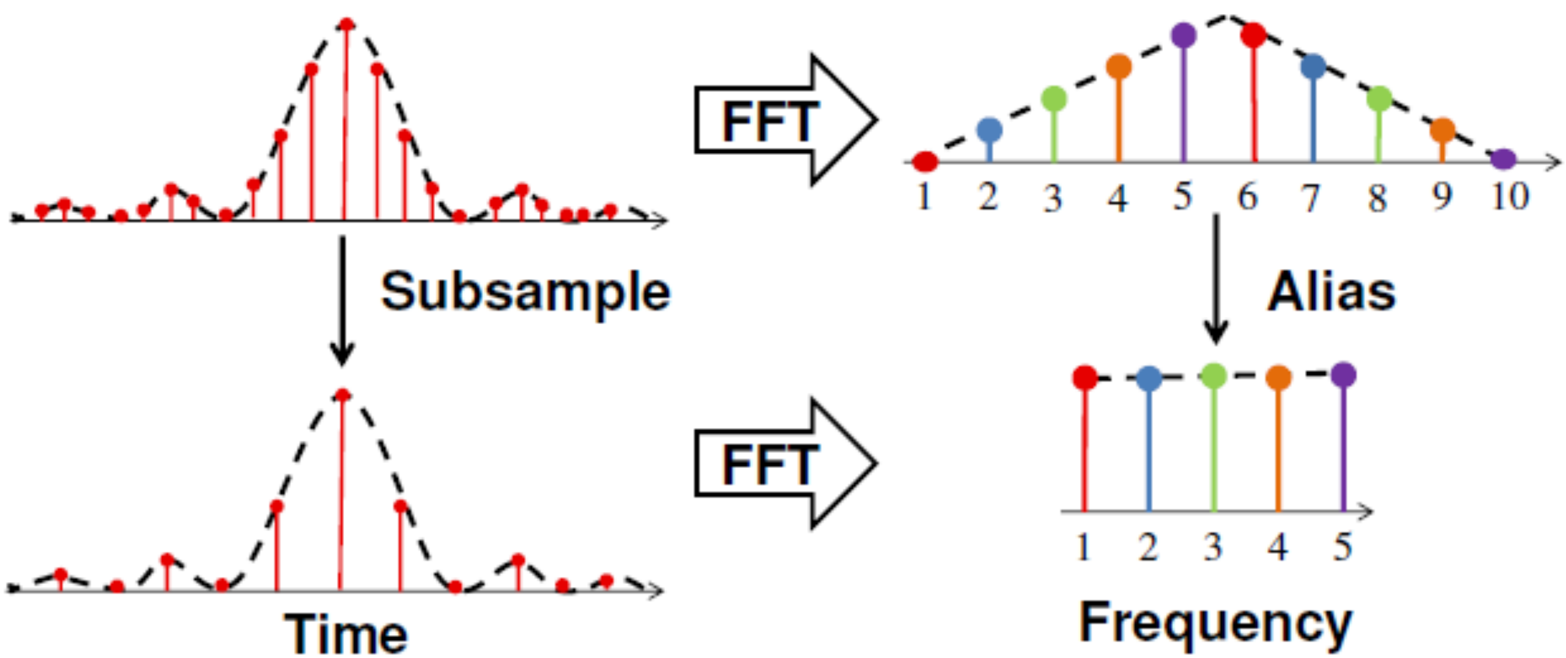




- Spike train is better (optimal): $|\mathrm{supp}(F)| * |\mathrm{supp}(F\hat{})| = n$ (see Mark Iwen's talk)
- However, can't hash multiple times
  - Frequencies i and j fall into the same bucket iff i=j mod B
  - Invariant under affine transform
- But what if the coefficients are distributed at random ? E.g., k-non-zeros in random places
  - Lawlor-Wang-Christlieb'12: O(k) samples, but sampling at arbitrary points

Bucketization using spike train ≈ aliasing

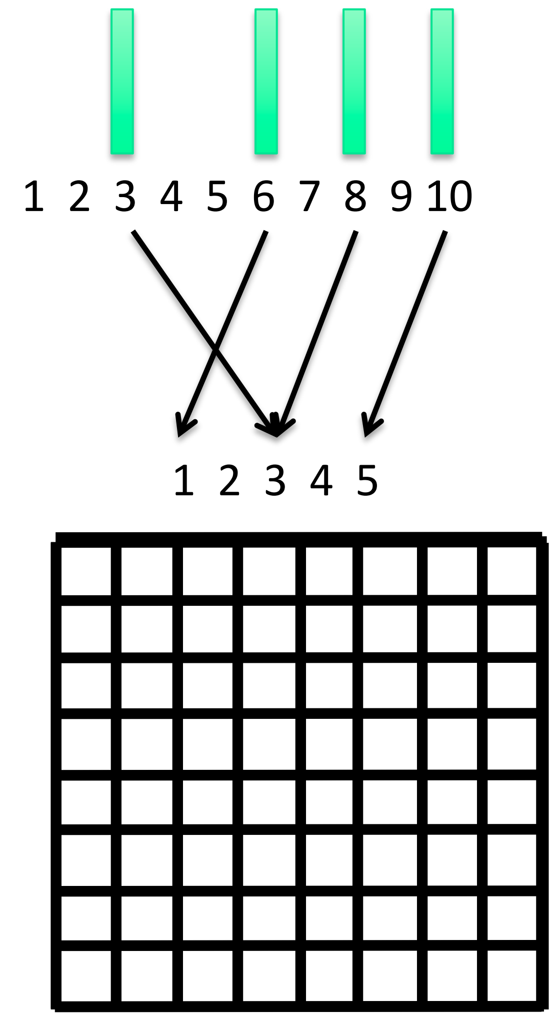
- Frequencies i and j fall into the same bucket iff i=j mod B



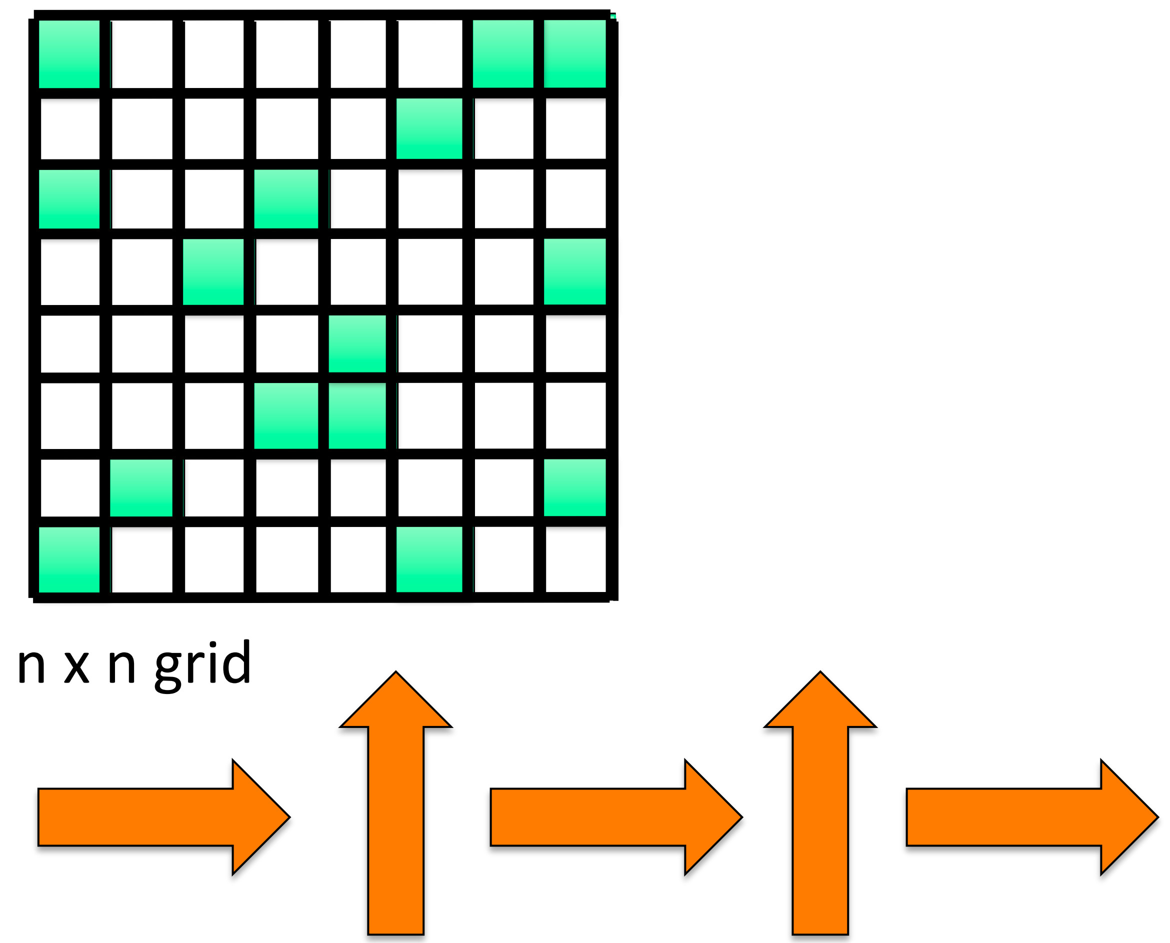
- What happens if we alias random frequencies ?

Spectral balls and bins

- k balls (coefficients), B buckets/bins
  - Need $B>k^2$ to achieve isolation
  - Sample complexity $>k^2$, i.e., bad
- Issue: we need more than one way of aliasing frequencies
- This is possible if, e.g., we have
  - 2D transform over na n x n grid
  - Suppose that $k \approx n$
- Then we can define buckets to be either single columns or single rows

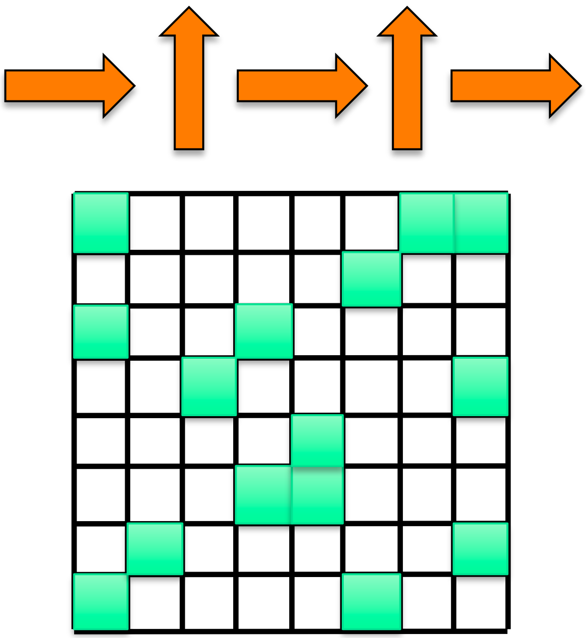


Alternating rows/columns



Analysis

- Analysis
  - With good probability (over the input) the process converges after O(log n) steps for k≈n
  - Each step requires O(n)=O(k) samples
  - But we can use the same samples in all steps (no rehashing)
  - So O(k) samples total
  - Time: O(k log n)
- Generalizes to
  - k<n : O(k) samples
  - l2/l2 recovery (Gaussian noise): O(k log n) samples



Conclusions

- Sparse FFT with running times $O(k \log n)$ or $O(k \log(n) \log(n/k))$
  - Improves over FFT for $k << n$
- Can improve sample complexity to $O(k)$ or $O(k \log n)$ in **2D average case**
  - Bonus: the algorithm is really simple
- Questions:
  - klogn time for **approximately** sparse signals?
    - Not clear: $k \log(n/k)$ samples needed, extra log n for FT
  - Better sample complexity in the **worst case**?
  - Deterministic ? (best known runtime $>k^2$))
  - Model-based (coefficients cluster in blocks) ?
  - …

Empirical evaluation: SFFT 3.0 (exact sparsity)