Sample-Optimal Average-Case Sparse Fourier Transform in Two Dimensions

Badih Ghazi Haitham Hassanieh Piotr Indyk
Dina Katabi Eric Price Lixin Shi

***Abstract*— We present the first sample-optimal sublinear time algorithms for the sparse Discrete Fourier Transform over a two-dimensional $\sqrt{n} \times \sqrt{n}$ grid. Our algorithms are analyzed for the average case signals. For signals whose spectrum is exactly sparse, we present algorithms that use $O(k)$ samples and run in $O(k \log k)$ time, where k is the expected sparsity of the signal. For signals whose spectrum is approximately sparse, we have an algorithm that uses $O(k \log n)$ samples and runs in $O(k \log^2 n)$ time, for $k = \Theta(\sqrt{n})$. All presented algorithms match the lower bounds on sample complexity for their respective signal models.**

I. Introduction

The Discrete Fourier Transform (DFT) is a powerful tool whose applications encompass video and audio processing [30], [12], [5], radar and GPS systems [13], [8], medical imaging, spectroscopy [19], [24], the processing of seismic data by the oil and gas industries [31], and many other engineering tasks. Currently, the fastest approach for computing the Discrete Fourier Transform uses the FFT algorithm. Given a signal of size n, the FFT computes its frequency representation in $O(n \log n)$ time. However, the emergence of big data problems, in which the processed datasets can exceed terabytes [27], has rendered the FFT's runtime too slow. Furthermore, in many domains (e.g., medical imaging [22], NMR spectroscopy [20]), data acquisition is costly or cumbersome, and hence one may be unable to collect enough measurements to compute the desired Fourier transform. These scenarios motivate the need for algorithms that compute the Fourier transform faster than the FFT, and use only a subset of the input data required by the FFT.

Recent efforts in the area of Fourier sampling have focused on addressing the above need. The resulting advances show that for sparse data (i.e., data that exhibits a limited number of dominating frequencies) one can design algorithms that operate only on a small subset of the input data, and run in sublinear time [23], [10], [2], [11], [17], [1], [15], [14], [21], [6], [13]. Since sparsity is common (in video, audio, medical imaging, NMR spectroscopy, GPS, seismic data, etc.), such results promise a significant impact on multiple application domains. The most efficient algorithms prior to this paper are given in [14], and offer the following performance guarantees:[1]

- For signals that are exactly k-sparse (i.e., signals that have exactly k nonzero Fourier coefficients), the algorithm runs in $O(k \log n)$ time.
- For approximately sparse signals, the algorithm runs in $O(k \log n \log(n/k))$ time, where k is the number of large Fourier coefficients.

While those past algorithms have achieved efficient running times, they suffer from important limitations. Perhaps the main limitation is that their sample complexity bounds are too high. In particular, the sample complexity of the exactly k-sparse algorithm is $\Theta(k \log n)$. This bound is suboptimal by a logarithmic factor, as it is known that one can recover any signal with k nonzero Fourier coefficients from $O(k)$ samples [3], albeit in super-linear time. The sample complexity of the approximately-sparse algorithm is $\Theta(k \log(n) \log(n/k))$. This bound is also a logarithmic factor away from the lower bound of $\Omega(k \log(n/k))$ [26].

Reducing the sample complexity is highly desirable as it typically implies a reduction in signal acquisition time, measurement overhead and communication cost. For example, in medical imaging the main goal is to reduce the sample complexity in order to reduce the time the patient spends in the MRI machine [22], or the radiation dose she receives [29]. Similarly in spectrum sensing, a lower average sampling rate enables the fabrication of efficient analog to digital converters (ADCs) that can acquire very wideband multi-GHz signals [32]. In fact, the central goal of the area of compressed sensing is to reduce the sample complexity.

A second limitation of the prior algorithms is that most

Badih Ghazi, Haitham Hassanieh, Piotr Indyk, Dina Katabi, Eric Price, and Lixin Shi are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA {badih, haithamh, indyk, dk, ecprice, lixshi}@mit.edu

[1]The Related Work section discusses additional algorithms that have different assumptions and guarantees.

of them are designed for one-dimensional signals. This is unfortunate, since multi-dimensional instances of DFT are often particularly sparse. This situation is somewhat alleviated by the fact that the two-dimensional DFT over $p \times q$ grids can be reduced to the one-dimensional DFT over a signal of length pq [11], [16]. However, the reduction applies only if p and q are relatively prime, which excludes the most typical case of $m \times m$ grids where m is a power of 2. The only prior algorithm that applies to general $m \times m$ grids, due to [11], has $O(k \log^c n)$ sample and time complexity for a rather large value of c. If n is a power of 2, a two-dimensional adaptation of the [15] algorithm (outlined in [9]) has roughly $O(k \log^3 n)$ time and sample complexity.

A. Our Results

In this paper, we present the first sample-optimal sublinear time algorithms for the Discrete Fourier Transform over a two-dimensional $\sqrt{n} \times \sqrt{n}$ grid. Our algorithms are analyzed in the average case. Our input distributions are natural. For the exactly sparse case, we assume the Bernoulli model: each spectrum coordinate is nonzero with probability k/n, in which case the entry assumes an arbitrary value predetermined for that position.[2] For the approximately-sparse case, we assume that the spectrum $\widehat{x}$ of the signal is a sum of two vectors: the signal vector, chosen from the Bernoulli distribution, and the noise vector, chosen from the Gaussian distribution (see Section §III Preliminaries for the complete definition). These or similar[3] distributions are often used as test cases for empirical evaluations of sparse Fourier Transform algorithms [18], [15], [21] or theoretical analysis of their performance [21].

The algorithms succeed with a constant probability. The notion of success depends on the scenario considered. For the exactly sparse case, an algorithm is successful if it recovers the spectrum exactly. For the approximately sparse case, the algorithm is successful if it reports a signal with spectrum $\widehat{z}$ such that:

$$\|\widehat{z} - \widehat{x}\|_2^2 = O(\sigma^2 n) + \|\widehat{x}\|_2^2 / n^c, \qquad (1)$$

where σ^2 denotes the variance of the normal distributions defining each coordinate of the noise vector, and where c is any constant. Note that any k-sparse approximation to $\widehat{x}$ has error $\Omega(\sigma^2 n)$ with overwhelming probability, and that the second term in the bound in Equation 1 is subsumed by the first term as long as the signal-to-noise ratio is at most polynomial, i.e., $\|\widehat{x}\|_2 \leq n^{O(1)}\sigma$. See Section §III for further discussion.

The running time and the sample complexity bounds of our algorithms are depicted in the following table (assuming $\sqrt{n}$ is a power of 2):

Alg.	Input	Samples	Time	Assumptions
1	Sparse	k	$k \log k$	$k = O(\sqrt{n})$
2	Sparse	$k+$ $k(\log\log n)^{O(1)}$	$k \log k$	Any k
3	Approx. sparse	$k \log n$	$k \log^2 n$	$k = \Theta(\sqrt{n})$

The key feature of our algorithms is that their sample complexity bounds are optimal. For the exactly sparse case, the lower bound of $\Omega(k)$ is immediate. For the approximately sparse case, we note that the $\Omega(k \log(n/k))$ lower bound of [26] holds even if the spectrum is the sum of a k-sparse signal vector in $\{0, 1, -1\}^n$ and Gaussian noise. The latter is essentially a special case of the distributions handled by our algorithm as shown in [9]. From the running time perspective, our algorithms are slightly faster than those in [14], with the improvement occurring for low values of k.

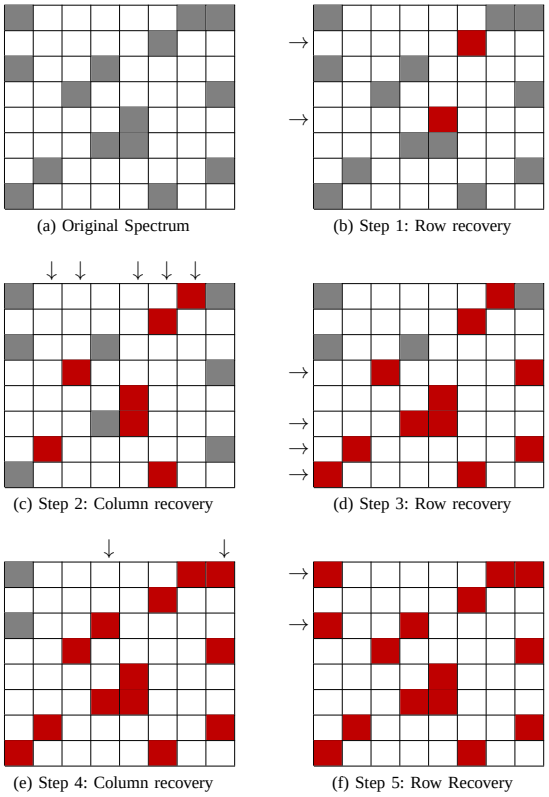
An additional feature of the first algorithm (in the table) is its simplicity and therefore its low "big-Oh" overhead. As a result, this algorithm is easy to adapt for practical applications. In [28], we have customized this algorithm and applied it to 2D Magnetic Resonance Spectroscopy (MRS). MRS is an advanced type of medical imaging used to detect biomarkers of diseases [4]. In this particular application, our algorithm outperformed compressive sensing and reduced the required measurements by almost a factor of 3x, hence reducing the overall cost and the time the patient has to spend in the MRI machine.

B. Our Techniques

Our first algorithm for k-sparse signals is based on the following observation: The spike-train filter (i.e., uniform sub-sampling) is one of the most efficient ways for mapping the Fourier coefficients into buckets. For one-dimensional signals however, this filter is not amenable to randomization. Hence, when multiple nonzero Fourier coefficients collide into the same bucket, one cannot efficiently resolve the collisions by randomizing the spike-train filter. In contrast, for two-dimensional signals, we naturally obtain two distinct spike-train filters, which correspond to subsampling the columns and subsampling the rows. Hence, we can resolve colliding nonzero Fourier coefficients by alternating between these two filters.

[2]Note that this model subsumes the scenario where the values of the nonzero coordinates are chosen i.i.d. from some distribution.

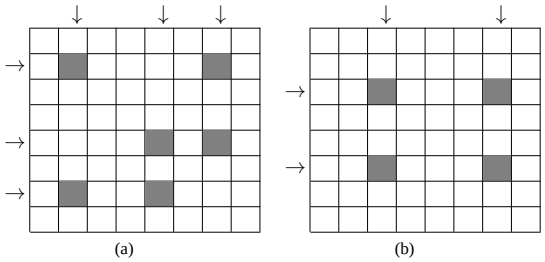
[3]A popular alternative is to use the hypergeometric distribution over the set of nonzero entries instead of the Bernoulli distribution. The advantage of the former is that it yields vectors of sparsity *exactly* equal to k. In this paper we opted for the Bernoulli model since it is simpler to analyze. However, both models are quite similar. In particular, for large enough k, the actual sparsity of vectors in the Bernoulli model is sharply concentrated around k.



(a) Original Spectrum (b) Step 1: Row recovery (c) Step 2: Column recovery (d) Step 3: Row recovery (e) Step 4: Column recovery (f) Step 5: Row Recovery

Fig. 1: An illustration of the "peeling" recovery process on an 8×8 signal with 15 nonzero frequencies. In each step, the algorithm recovers all 1-sparse columns and rows (the recovered entries are depicted in red). The process converges after a few steps.

More specifically, recall that one way to compute the two-dimensional DFT of a signal x is to apply the one-dimensional DFT to each column and then to each row. Suppose that $k = a\sqrt{n}$ for $a < 1$. In this case, the expected number of nonzero entries in each row is less than 1. If *every* row contained exactly one nonzero entry, then the DFT could be computed via the following two step process. In the first step, we select the first two columns of x, denoted by $u^{(0)}$ and $u^{(1)}$, and compute their DFTs $\widehat{u}^{(0)}$ and $\widehat{u}^{(1)}$. Let j_i be the index of the unique nonzero entry in the i-th row of $\widehat{x}$, and let a be its value. Observe that $\widehat{u}_i^{(0)} = a$ and $\widehat{u}_i^{(1)} = a\omega^{-j_i}$ (where ω is a primitive $\sqrt{n}$-th root of unity), as these are the first two entries of the inverse Fourier transform of a 1-sparse signal ae_{j_i}. Thus, in the second step, we can retrieve the value of the nonzero entry, equal to $\widehat{u}_i^{(0)}$, as well as the index j_i from the phase of the ratio $\widehat{u}_i^{(1)}/\widehat{u}_i^{(0)}$. (this technique was introduced in [14], [21] and was referred to as the "OFDM trick"). The total time is dominated by the cost of the two DFTs of the columns, which is $O(\sqrt{n}\log n)$. Since the algorithm queries only a constant number of columns, its sample complexity is $O(\sqrt{n})$.



(a) (b)

Fig. 2: Examples of obstructing sequences of nonzero entries. None of the remaining rows or columns has a sparsity of 1.

In general, the distribution of the nonzero entries over the rows can be non-uniform –i.e., some rows may have multiple nonzero Fourier coefficients. Thus, our actual algorithm alternates the above recovery process between the columns and rows (see Figure 1 for an illustration). Since the OFDM trick works only on 1-sparse columns/rows, we check the 1-sparsity of each column/row by sampling a constant number of additional entries. We then show that, as long as the sparsity constant a is small enough, this process recovers all entries in a logarithmic number steps with constant probability. The proof uses the fact that the probability of the existence of an "obstructing configuration" of nonzero entries which makes the process deadlocked (e.g., see Figure 2) is upper bounded by a small constant.

The algorithm is extended to the case of $k = o(\sqrt{n})$ via a reduction. Specifically, we subsample the signal x by the reduction ratio $R = \alpha\sqrt{n}/k$ for some small enough constant α in each dimension. The subsampled signal x' has dimension $\sqrt{m} \times \sqrt{m}$, where $\sqrt{m} = \frac{k}{\alpha}$. Since subsampling in time domain corresponds to "spectrum folding", i.e., adding together all frequencies with indices that are equal modulo $\sqrt{m}$, the nonzero entries of $\widehat{x}$ are mapped into the entries of $\widehat{x'}$. It can be seen that, with constant probability, the mapping is one-to-one. If this is the case, we can use the earlier algorithm for sparse DFT to compute the nonzero frequencies in $O(\sqrt{m}\log m) = O(\sqrt{k}\log k)$ time, using $O(k)$ samples. We then use the OFDM trick to identify the positions of those frequencies in $\widehat{x}$.

Our second algorithm for the exactly sparse case works for all values of k. The main idea behind it is to decode rows/columns with higher sparsity than 1. First,

we give a *deterministic*, *worst-case* algorithm for 1-dimensional sparse Fourier transforms that takes $O(k^2 + k(\log\log n)^{O(1)})$ time. This algorithm uses the relationship between sparse recovery and syndrome decoding of Reed-Solomon codes (due to [3]). Although a simple application of the decoder yields $O(n^2)$ decoding time, we show that by using appropriate numerical subroutines one can in fact recover a k-sparse vector from $O(k)$ samples in time $O(k^2 + k(\log\log n)^{O(1)})$.[4] In particular, we use Berlekamp-Massey's algorithm for constructing the error-locator polynomial and Pan's algorithm for finding its roots. For our fast average-case, 2-dimensional sparse Fourier transform algorithm, we fold the spectrum into $B = \frac{k}{C \log k}$ bins for some large constant C. Since the positions of the k nonzero frequencies are random, it follows that each bin receives $t = \Theta(\log k)$ frequencies with high probability. We then take $\Theta(t)$ samples of the time domain signal corresponding to each bin, and recover the frequencies corresponding to those bins in $O(t^2 + t(\log\log n)^{O(1)})$ time per bin, for a total time of $O(k \log k + k(\log\log n)^{O(1)})$.

This approach works as long as the number of nonzero coefficients per column/row are highly concentrated. However, this is not the case for $k \ll \sqrt{n} \log n$. We overcome this difficulty by replacing a row by a sequence of rows. A technical difficulty is that the process might lead to collisions of coefficients. We resolve this issue by using a two level procedure, where the first level returns the syndromes of colliding coefficients as opposed to the coefficients themselves; the syndromes are then decoded at the second level.

The above description summarizes the second algorithm. Due to space limitations, the full-description of the second algorithm along with the proofs of some lemmas are not included in this paper. They are however available on arxiv [9].

Our third algorithm works for *approximately* sparse data, at sparsity $\Theta(\sqrt{n})$. Its general outline mimics that of the first algorithm. Specifically, it alternates between decoding columns and rows, assuming that they are 1-sparse. The decoding subroutine itself is similar to that of [14] and uses $O(\log n)$ samples. The subroutine first checks whether the decoded entry is large; if not, the spectrum is unlikely to contain any large entry, and the subroutine terminates. The algorithm then subtracts the decoded entry from the column and checks whether the resulting signal contains no large entries in the spectrum (which would be the case if the original spectrum was approximately 1-sparse and the decoding was successful). The check is done by sampling $O(\log n)$ coordinates and checking whether their sum of squares is small. To prove that this check works with high probability, we use the fact that a collection of random rows of the Fourier matrix is likely to satisfy the Restricted Isometry Property (RIP) of [7].

A technical difficulty in the analysis of the algorithm is that the noise accumulates in successive iterations. This means that a $1/\log^{O(1)} n$ fraction of the steps of the algorithm will fail. However, we show that the dependencies are "local", which means that our analysis still applies to a vast majority of the recovered entries. We continue the iterative decoding for $\log\log n$ steps, which ensures that all but a $1/\log^{O(1)} n$ fraction of the large frequencies are correctly recovered. To recover the remaining frequencies, we resort to algorithms with worst-case guarantees.

[4]We note that, for $k = o(\log n)$, this is the fastest known *worst-case* algorithm for the exactly sparse DFT.

C. Extensions

Our algorithms have natural extensions to dimensions higher than 2. We do not include them in this paper as the description and analysis are rather cumbersome.

Moreover, due to the equivalence between the two-dimensional case and the one-dimensional case where n is a product of different prime powers [11], [16], our algorithms also give optimal sample complexity bounds for such values of n (e.g., $n = 6^t$) in the average case.

II. Related Work

As described in the introduction, the most efficient prior algorithms for computing the sparse DFT are due to [14]. For signals that are exactly k-sparse, the first algorithm runs in $O(k \log n)$ time. For approximately sparse signals, the second algorithm runs in $O(k \log n \log(n/k))$ time. Formally, the latter algorithm works for any signal x, and computes an approximation vector $\widehat{x}'$ that satisfies the ℓ_2/ℓ_2 approximation guarantee, i.e., $\|\widehat{x} - \widehat{x}'\|_2 \leq C \min_{k\text{-sparse } y} \|\widehat{x} - y\|_2$, where C is some approximation factor and the minimization is over k-sparse signals. Note that this guarantee generalizes that of Equation (1).

After this work was completed (see the arxiv version [9]), we became aware that another group has concurrently developed an efficient algorithm for the one dimensional exactly k-sparse case where the size of the signal n is product of primes (i.e., can be formed as a 2D discrete Fourier transform problem) [25]. Their algorithm is analyzed for the average case and achieves $O(k)$ sample complexity and runs in $O(k \log k)$. In comparison, our algorithms achieve similar guarantees for the exactly k-sparse case, but they further address the general case where the signal is contaminated by noise.

We also mention another efficient algorithm, due to [21], designed for the exactly k-sparse model. The

average case analysis presented in that paper also shows that the algorithm has $O(k)$ expected sample complexity and runs in $O(k \log k)$ time. However, the algorithm assumes the input signal x is specified as a *function* over an interval $[0, 1]$ that can be sampled at arbitrary positions, as opposed to a given discrete sequence of n samples as in our case. Thus, though very efficient, that algorithm does not solve the Discrete Fourier Transform problem.

III. Preliminaries

This section introduces the notation, assumptions and definitions used in the rest of this paper.

A. Notation: Throughout the paper we assume that $\sqrt{n}$ is a power of 2. We use $[m]$ to denote the set $\{0, \ldots, m-1\}$, and $[m] \times [m] = [m]^2$ to denote the $m \times m$ grid $\{(i, j) : i \in [m], j \in [m]\}$. We define $\omega = e^{-2\pi \mathbf{i}/\sqrt{n}}$ to be a primitive $\sqrt{n}$-th root of unity and $\omega' = e^{-2\pi \mathbf{i}/n}$ to be a primitive n-th root of unity. For any complex number a, we use $\phi(a) \in [0, 2\pi)$ to denote the *phase* of a. For a 2D matrix $x \in \mathbb{C}^{\sqrt{n} \times \sqrt{n}}$, its support is denoted by $\mathrm{supp}(x) \subseteq [\sqrt{n}] \times [\sqrt{n}]$. We use $\|x\|_0$ to denote $|\mathrm{supp}(x)|$, the number of nonzero coordinates of x. Its 2D Fourier spectrum is denoted by $\widehat{x}$, with

$$\widehat{x}_{i,j} = \frac{1}{\sqrt{n}} \sum_{l \in [\sqrt{n}]} \sum_{m \in [\sqrt{n}]} \omega^{il+jm} x_{l,m}$$

Similarly, if y is a frequency-domain signal, its inverse Fourier transform is denoted by $\check{y}$.

B. Definitions: The paper uses the comb filter used in [17], [15] (cf. [23]). The filter can be generalized to 2 dimensions as follows:

Given $(\tau_r, \tau_c) \in [\sqrt{n}] \times [\sqrt{n}]$, and B_r, B_c that divide $\sqrt{n}$, then for all $(i, j) \in [B_r] \times [B_c]$ set

$$y_{i,j} = x_{i(\sqrt{n}/B_r)+\tau_r, j(\sqrt{n}/B_c)+\tau_c}$$

Then, compute the 2D DFT $\hat{y}$ of y. Observe that $\hat{y}$ is a folded version of $\hat{x}$:

$$\hat{y}_{i,j} = \sum_{l \in [\frac{\sqrt{n}}{B_r}]} \sum_{m \in [\frac{\sqrt{n}}{B_c}]} \hat{x}_{lB_r+i, mB_c+j} \omega^{-\tau_r(i+lB_r)-\tau_c(j+mB_c)}$$

C. Distributions: In the exactly sparse case, we assume a Bernoulli model for the support of $\widehat{x}$. This means that for all $(i, j) \in [\sqrt{n}] \times [\sqrt{n}]$,

$$\Pr\{(i, j) \in \mathrm{supp}\,(\widehat{x})\} = k/n$$

and thus $\mathbb{E}[|\mathrm{supp}\,(\widehat{x})|] = k$. We assume an unknown predefined matrix $a_{i,j}$ of values in $\mathbb{C}$; if $\widehat{x}_{i,j}$ is selected to be nonzero, its value is set to $a_{i,j}$.

In the approximately sparse case, we assume that the signal $\widehat{x}$ is equal to $\widehat{x^*} + \widehat{w} \in \mathbb{C}^{\sqrt{n} \times \sqrt{n}}$, where $\widehat{x^*}_{i,j}$ is the "signal" and $\widehat{w}$ is the "noise". In particular, $\widehat{x^*}$ is drawn from the Bernoulli model, where $\widehat{x^*}_{i,j}$ is drawn from $\{0, a_{i,j}\}$ at random independently for each (i, j) for some values $a_{i,j}$ and with $\mathbb{E}[|\mathrm{supp}(\widehat{x^*})|] = k$. We also require that $|a_{i,j}| \geq L$ for some parameter L. $\widehat{w}$ is a complex Gaussian vector with variance σ^2 in both the real and imaginary axes independently on each coordinate; we notate this as $\widehat{w} \sim N_{\mathbb{C}}(0, \sigma^2 I_n)$. We will need that $L = C\sigma\sqrt{n/k}$ for a sufficiently large constant C, so that $\mathbb{E}[\|\widehat{x^*}\|_2^2] \geq C\,\mathbb{E}[\|\widehat{w}\|_2^2]$.

IV. Basic Algorithm for the Exactly Sparse Case

The algorithm for the noiseless case depends on the sparsity k where $k = \mathbb{E}[|\mathrm{supp}\,(\widehat{x})|]$ for a Bernoulli distribution of the support.

A. Basic Exact Algorithm: $k = \Theta(\sqrt{n})$

In this section, we focus on the regime $k = \Theta(\sqrt{n})$. Specifically, we will assume that $k = a\sqrt{n}$ for a (sufficiently small) constant $a > 0$.

The algorithm BasicExact2DSFFT is described as Algorithm IV.1. The key idea is to fold the spectrum into bins using the comb filter defined in §III and estimate frequencies which are isolated in a bin. The algorithm takes the FFT of a row and as a result frequencies in the same columns will get folded into the same row bin. It also takes the FFT of a column and consequently frequencies in the same rows wil get folded into the same column bin. The algorithm then uses the OFDM trick introduced in [14] to recover the columns and rows whose sparsity is 1. It iterates between the column bins and row bins, subtracting the recovered frequencies and estimating the remaining columns and rows whose sparsity is 1. An illustration of the algorithm running on an 8×8 signal with 15 nonzero frequencies is shown in Fig. 1 in Section §I. The algorithm also takes a constant number of extra FFTs of columns and rows to check for collisions within a bin and avoid errors resulting from estimating bins where the sparsity is greater than 1. The algorithm uses three functions:

- FoldToBins. This procedure folds the spectrum into $B_r \times B_c$ bins using the comb filter described §III.
- BasicEstFreq. Given the FFT of rows or columns, it estimates the frequency in the large bins. If there is no collision, i.e., if there is a single nonzero frequency in the bin, it adds this frequency to the result $\widehat{w}$ and subtracts its contribution to the row and column bins.
- BasicExact2DSFFT. This performs the FFT of the rows and columns and then iterates BasicEst-

procedure FOLDTOBINS(x, B_r, B_c, τ_r, τ_c)
  $y_{i,j} = x_{i(\sqrt{n}/B_r)+\tau_r, j(\sqrt{n}/B_c)+\tau_c}$ for $(i,j) \in [B_r] \times [B_c]$,
  return $\widehat{y}$, the DFT of y
procedure BASICESTFREQ($\widehat{u}^{(T)}$, $\widehat{v}^{(T)}$,T, IsCol)
  $\widehat{w} \leftarrow 0$.
  Compute $J = \{j : \sum_{\tau \in T} |\widehat{u}_j^{(\tau)}| > 0\}$.
  for $j \in J$ **do**
    $b \leftarrow \widehat{u}_j^{(1)} / \widehat{u}_j^{(0)}$.
    $i \leftarrow \text{round}(\phi(b) \frac{\sqrt{n}}{2\pi}) \bmod \sqrt{n}$. ▷ $\phi(b)$ is the phase of b.
    $s \leftarrow \widehat{u}_j^{(0)}$.
      ▷ Test whether the row or column is 1-sparse
    if $\left(\sum_{\tau \in T} |\widehat{u}_j^{(\tau)} - s\omega^{-\tau i}| == 0\right)$ **then**
      if IsCol **then** ▷ whether decoding column or row
        $\widehat{w}_{i,j} \leftarrow s$.
      else
        $\widehat{w}_{j,i} \leftarrow s$.
      for $\tau \in T$ **do**
        $\widehat{u}_j^{(\tau)} \leftarrow 0$
        $\widehat{v}_i^{(\tau)} \leftarrow \widehat{v}_i^{(\tau)} - s\omega^{-\tau i}$
  return $\widehat{w}$, $\widehat{u}^{(T)}$, $\widehat{v}^{(T)}$
procedure BASICEXACT2DSFFT(x, k)
  $T \leftarrow [2c]$ ▷ We set $c \geq 6$
  for $\tau \in T$ **do**
    $\widehat{u}^{(\tau)} \leftarrow$ FOLDTOBINS$(x, \sqrt{n}, 1, 0, \tau)$.
    $\widehat{v}^{(\tau)} \leftarrow$ FOLDTOBINS$(x, 1, \sqrt{n}, \tau, 0)$.
  $\widehat{z} \leftarrow 0$
  for $t \in [C \log n]$ **do** ▷ $\widehat{u}^{(T)} := \{\widehat{u}^{(\tau)} : \tau \in T\}$
    $\{\widehat{w}, \widehat{u}^{(T)}, \widehat{v}^{(T)}\} \leftarrow$ BASICESTFREQ$(\widehat{u}^{(T)}, \widehat{v}^{(T)}, T,$ true).
    $\widehat{z} \leftarrow \widehat{z} + \widehat{w}$.
    $\{\widehat{w}, \widehat{v}^{(T)}, \widehat{u}^{(T)}\} \leftarrow$ BASICESTFREQ$(\widehat{v}^{(T)}, \widehat{u}^{(T)}, T,$ false).
    $\widehat{z} \leftarrow \widehat{z} + \widehat{w}$.
  return $\widehat{z}$

Algorithm IV.1: Basic Exact 2D sparse FFT algorithm for $k = \Theta(\sqrt{n})$

FREQ between the rows and columns until is recovers $\widehat{x}$.

Analysis of BASICEXACT2DSFFT*:* The analysis relies on the following two lemmas which we prove in [9].

Lemma 4.1: For any constant $\alpha > 0$, if $a > 0$ is a sufficiently small constant, then assuming that all 1-sparsity tests in the procedure BASICESTFREQ are correct, the algorithm reports the correct output with probability at least $1 - O(\alpha)$.

Lemma 4.2: The probability that any 1-sparsity test invoked by the algorithm is incorrect is at most $O(1/n^{(c-5)/2})$.

Theorem 4.3: For any constant α, the algorithm BASICEXACT2DSFFT uses $O(\sqrt{n})$ samples, runs in time $O(\sqrt{n} \log n)$ and returns the correct vector $\widehat{x}$ with probablility at least $1 - O(\alpha)$ as long as a is a small enough constant.

Proof: From Lemma 4.1 and Lemma 4.2, the algorithm returns the correct vector $\widehat{x}$ with probability at least $1 - O(\alpha) - O(n^{-(c-5)/2}) = 1 - O(\alpha)$ for $c > 5$.

The algorithm uses only $O(T) = O(1)$ rows and columns of x, which yields $O(\sqrt{n})$ samples. The running time is bounded by the time needed to perform $O(1)$ FFTs of rows and columns (in FOLDTOBINS) procedure, and $O(\log n)$ invocations of BASICESTFREQ. Both components take time $O(\sqrt{n} \log n)$.

■

B. Reduction to Basic Exact Algorithm: $k = o(\sqrt{n})$

Algorithm REDUCEEXACT2DSFFT, which is for the case where $k = o(\sqrt{n})$, is described in Algorithm IV.2. The key idea is to reduce the problem from the case where $k = o(\sqrt{n})$ to the case where $k = \Theta(\sqrt{n})$. To do that, we subsample the input time domain signal x by the reduction ratio $R = a\sqrt{n}/k$ for some small enough a. The subsampled signal x' has dimension $\sqrt{m} \times \sqrt{m}$, where $\sqrt{m} = \frac{k}{a}$. This implies that the probability that any coefficient in $\widehat{x}'$ is nonzero is at most $R^2 \times k/n = a^2/k = (a^2/k) \times (k^2/a^2)/m = k/m$, since $m = k^2/a^2$. This means that we can use the algorithm BASICNOISELESS2DSFFT in subsection §IV-A to recover $\widehat{x}'$. Each of the entries of $\widehat{x}'$ is a frequency in $\widehat{x}$ which was folded into $\widehat{x}'$. We employ the same phase technique used in [14] and subsection §IV-A to recover their original frequency position in $\widehat{x}$.

The algorithm uses two functions:

- REDUCETOBASICSFFT: This folds the spectrum into $O(k) \times O(k)$ dimensions and performs the reduction to BASICEXACT2DSFFT. Note that only the $O(k)$ elements of x' which will be used in BASICEXACT2DSFFT need to be computed.
- REDUCEEXACT2DSFFT: This invokes the reduction as well as the phase technique to recover $\widehat{x}$.

Analysis of REDUCEEXACT2DSFFT*:* We state the following lemma which we prove in [9].

Lemma 4.4: For any constant α, for sufficiently small a there is a one-to-one mapping of frequency coefficients from $\widehat{x}$ to $\widehat{x}'$ with probability at least $1 - \alpha$.

Theorem 4.5: For any constant $\alpha > 0$, there exists a constant $c > 0$ such that if $k < c\sqrt{n}$ then the

procedure REDUCETOBASICSFFT(x, R, τ_r, τ_c)
  Define $x'_{ij} = x_{iR+\tau_r, jR+\tau_c}$ ▷ With lazy evaluation
  return BASICEXACT2DSFFT(x', k)
procedure REDUCEEXACT2DSFFT(x, k)
  $R \leftarrow \frac{a\sqrt{n}}{k}$, for some constant $a < 1$ such that $R|\sqrt{n}$.
  $\widehat{u}^{(0,0)} \leftarrow$ REDUCETOBASICSFFT$(x, R, 0, 0)$
  $\widehat{u}^{(1,0)} \leftarrow$ REDUCETOBASICSFFT$(x, R, 1, 0)$
  $\widehat{u}^{(0,1)} \leftarrow$ REDUCETOBASICSFFT$(x, R, 0, 1)$
  $\widehat{z} \leftarrow 0$
  $L \leftarrow \text{supp}(\widehat{u}^{(0,0)}) \cap \text{supp}(\widehat{u}^{(1,0)}) \cap \text{supp}(\widehat{u}^{(0,1)})$
  for $(\ell, m) \in L$ **do**
    $b_r \leftarrow \widehat{u}^{(1,0)}_{\ell,m} / \widehat{u}^{(0,0)}_{\ell,m}$
    $i \leftarrow \text{round}(\phi(b_r)\frac{\sqrt{n}}{2\pi}) \bmod \sqrt{n}$
    $b_c \leftarrow \widehat{u}^{(0,1)}_{\ell,m} / \widehat{u}^{(0,0)}_{\ell,m}$
    $j \leftarrow \text{round}(\phi(b_c)\frac{\sqrt{n}}{2\pi}) \bmod \sqrt{n}$
    $\widehat{z}_{ij} \leftarrow \widehat{u}^{(0,0)}_{\ell,m}$
  return $\widehat{z}$

Algorithm IV.2: Exact 2D sparse FFT algorithm for $k = o(\sqrt{n})$

algorithm REDUCEEXACT2DSFFT uses $O(k)$ samples, runs in time $O(k \log k)$ and returns the correct vector $\widehat{x}$ with probablility at least $1 - \alpha$.

Proof: By Theorem 4.3 and the fact that each coefficient in $\widehat{x}'$ is nonzero with probability $O(1/k)$, each invocation of the function REDUCETOBASICSFFT fails with probability at most α. By Lemma 4.4, with probability at least $1 - \alpha$, we could recover $\widehat{x}$ correctly if each of the calls to REDTOBASICSFFT returns the correct result. By the union bound, the algorithm REDUCEEXACT2DSFFT fails with probability at most $\alpha + 3 \times \alpha = O(\alpha)$.

The algorithm uses $O(1)$ invocations of BASICEXACT2DSFFT on a signal of size $O(k) \times O(k)$ in addition to $O(k)$ time to recover the support using the OFDM trick. Noting that calculating the intersection L of supports takes $O(k)$ time, the stated number of samples and running time then follow directly from Theorem 4.3. ■

V. ALGORITHM FOR ROBUST RECOVERY

The algorithm for noisy recovery ROBUST2DSFFT is shown in Algorithm V.1. The algorithm is very similar to the exactly sparse case. It first takes FFT of rows and columns using FOLDTOBINS procedure. It then iterates between the columns and rows, recovering frequencies in bins which are 1-sparse using the ROBUSTESTIMATECOL procedure. This procedure uses the function HIKPLOCATESIGNAL from [14] to make the estimation of the frequency positions robust to noise.

procedure ROBUSTESTIMATECOL($\widehat{u}$, $\widehat{v}$, T, T', IsCol, J, Ranks)
  $\widehat{w} \leftarrow 0$.
  $S \leftarrow \{\}$ ▷ Set of changes, to be tested next round.
  for $j \in J$ **do**
    continue if Ranks[(IsCol, j)] $\geq \log \log n$.
    $i \leftarrow$ HIKPLOCATESIGNAL$(\widehat{u}^{(T')}, T')$
      ▷ Procedure from [14]: $O(\log^2 n)$ time
    $a \leftarrow \text{median}_{\tau \in T}\, \widehat{u}^{\tau}_j \omega^{\tau i}$.
    continue if $|a| < L/2$
      ▷ Nothing significant recovered
    continue if $\sum_{\tau \in T} |\widehat{u}^{\tau}_j - a\omega^{-\tau i}|^2 \geq L^2 |T| / 10$
      ▷ Bad recovery: probably not 1-sparse
    $b \leftarrow \text{mean}_{\tau \in T}\, \widehat{u}^{\tau}_j \omega^{\tau i}$.
    if IsCol **then** ▷ whether decoding column or row
      $\widehat{w}_{i,j} \leftarrow b$.
    else
      $\widehat{w}_{j,i} \leftarrow b$.
    $S \leftarrow S \cup \{i\}$.
    Ranks[(1 − IsCol, i)] += Ranks[(IsCol, j)].
    for $\tau \in T \cup T'$ **do**
      $\widehat{u}^{(\tau)}_j \leftarrow \widehat{u}^{(\tau)}_j - b\omega^{-\tau i}$
      $\widehat{v}^{(\tau)}_i \leftarrow \widehat{v}^{(\tau)}_i - b\omega^{-\tau i}$
  return $\widehat{w}$, $\widehat{u}$, $\widehat{v}$, S
procedure ROBUST2DSFFT(x, k)
  $T, T' \subset [\sqrt{n}], |T| = |T'| = O(\log n)$
  for $\tau \in T \cup T'$ **do**
    $\widehat{u}^{(\tau)} \leftarrow$ FOLDTOBINS$(x, \sqrt{n}, 1, 0, \tau)$.
    $\widehat{v}^{(\tau)} \leftarrow$ FOLDTOBINS$(x, 1, \sqrt{n}, \tau, 0)$.
  $\widehat{z} \leftarrow 0$
  $R \leftarrow 1^{[2] \times [\sqrt{n}]}$ ▷ Rank of vertex (iscolumn, index)
  $S_{col} \leftarrow [\sqrt{n}]$ ▷ Which columns to test
  for $t \in [C \log n]$ **do**
    $\{\widehat{w}, \widehat{u}, \widehat{v}, S_{row}\} \leftarrow$
      ROBUSTESTIMATECOL$(\widehat{u}, \widehat{v}, T, T', \text{true}, S_{col}, R)$.
    $\widehat{z} \leftarrow \widehat{z} + \widehat{w}$.
    $S_{row} \leftarrow [\sqrt{n}]$ if $t = 0$ ▷ Try each row the first time
    $\{\widehat{w}, \widehat{v}, \widehat{u}, S_{col}\} \leftarrow$
      ROBUSTESTIMATECOL$(\widehat{v}, \widehat{u}, T, T', \text{false}, S_{row}, R)$.
    $\widehat{z} \leftarrow \widehat{z} + \widehat{w}$.
  return $\widehat{z}$

Algorithm V.1: Robust 2D sparse FFT algorithm for $k = \Theta(\sqrt{n})$

A. Analysis of Each Stage of Recovery

Here, we show that each step of the recovery is correct with high probability using the following two lemmas. The first lemma shows that with very low probability the ROBUSTESTIMATECOL procedure generates a false

negative (misses a frequency), false positive (adds a fake frequency) or a bad update (wrong estimate of a frequency). The second lemma is analogus to lemma 4.2 and shows that the probability that the 1-sparse test fails when there is noise is low. The proof of these lemmas is long and can be found in [9].

Lemma 5.1: Consider the recovery of a column/row j in ROBUSTESTIMATECOL, where $\widehat{u}$ and $\widehat{v}$ are the results of FOLDTOBINS on $\widehat{x}$. Let $y \in \mathbb{C}^{\sqrt{n}}$ denote the jth column/row of $\widehat{x}$. Suppose y is drawn from a permutation invariant distribution $y = y^{head} + y^{residue} + y^{gauss}$, where $\min_{i \in \mathrm{supp}(y^{head})} |y_i| \geq L$, $\|y^{residue}\|_1 < \epsilon L$, and y^{gauss} is drawn from the $\sqrt{n}$-dimensional normal distribution $N_{\mathbb{C}}(0, \sigma^2 I_{\sqrt{n}})$ with standard deviation $\sigma = \epsilon L / n^{1/4}$ in each coordinate on both real and imaginary axes. We do not require that y^{head}, $y^{residue}$, and y^{gauss} are independent except for the permutation invariance of their sum.

Consider the following bad events:

- False negative: $\mathrm{supp}(y^{head}) = \{i\}$ and ROBUSTESTIMATECOL does not update coordinate i.
- False positive: ROBUSTESTIMATECOL updates some coordinate i but $\mathrm{supp}(y^{head}) \neq \{i\}$.
- Bad update: $\mathrm{supp}(y^{head}) = \{i\}$ and coordinate i is estimated by b with $\left|b - y_i^{head}\right| > \|y^{residue}\|_1 + \sqrt{\frac{\log \log n}{\log n}} \epsilon L$.

For any constant c and ϵ below a sufficiently small constant, there exists a distribution over sets T, T' of size $O(\log n)$, such that as a distribution over y and T, T' we have

- The probability of a false negative is $1/\log^c n$.
- The probability of a false positive is $1/n^c$.
- The probability of a bad update is $1/\log^c n$.

Lemma 5.2: Let $y \in \mathbb{C}^m$ be drawn from a permutation invariant distribution with $r \geq 2$ nonzero values. Suppose that all the nonzero entries of y have absolute value at least L. Choose $T \subset [m]$ uniformly at random with $t := |T| = O(c^3 \log m)$.

Then, the probability that there exists a y' with $\|y'\|_0 \leq 1$ and

$$\|(\check{y} - \check{y}')_T\|_2^2 < \epsilon L^2 t / n$$

is at most $c^3 (\frac{c}{m-r})^{c-2}$ whenever $\epsilon < 1/8$.

B. Analysis of Overall Recovery

Recall that we are considering the recovery of a signal $\widehat{x} = \widehat{x^*} + \widehat{w} \in \mathbb{C}^{\sqrt{n} \times \sqrt{n}}$, where $\widehat{x^*}$ is drawn from the Bernoulli model with expected $k = a\sqrt{n}$ nonzero entries for a sufficiently small constant a, and $\widehat{w} \sim N_{\mathbb{C}}(0, \sigma^2 I_n)$ with $\sigma = \epsilon L \sqrt{k/n} = \Theta(\epsilon L / n^{1/4})$ for sufficiently small ϵ.

It will be useful to consider a bipartite graph representation G of $\widehat{x^*}$. We construct a bipartite graph with $\sqrt{n}$ nodes on each side, where the left side corresponds to rows and the right side corresponds to columns. For each $(i, j) \in \mathrm{supp}(\widehat{x^*})$, we place an edge between left node i and right node j of weight $\widehat{x^*}_{(i,j)}$.

Our algorithm is a "peeling" procedure on this graph. It iterates over the vertices, and can with a "good probability" recover an edge if it is the only incident edge on a vertex. Once the algorithm recovers an edge, it can remove it from the graph. The algorithm will look at the column vertices, then the row vertices, then repeat; these are referred to as *stages*. Supposing that the algorithm succeeds at recovery on each vertex, this gives a canonical order to the removal of edges. Call this the *ideal* ordering.

In the ideal ordering, an edge e is removed based on one of its incident vertices v. This happens after all other edges reachable from v without passing through e are removed. Define the *rank* of v to be the number of such reachable edges, and $\mathrm{rank}(e) = \mathrm{rank}(v) + 1$ (with $\mathrm{rank}(v)$ undefined if v is not used for recovery of any edge).

Lemma 5.3: Let c, α be arbitrary constants, and a a sufficiently small constant depending on c, α. Then with $1 - \alpha$ probability every component in G is a tree and at most $k/\log^c n$ edges have rank at least $\log \log n$.

Proof: Each edge of G appears independently with probability $k/n = a/\sqrt{n}$. There are at most $\sqrt{n}^t$ cycles of length t. Hence the probability that any cycle of length t exists is at most a^t, so the chance any cycle exists is less than $a^2/(1 - a^2) < \alpha/2$ for sufficiently small a.

Each vertex has expected degree $a < 1$. Exploring the component for any vertex v is then a subcritical branching process, so the probability that v's component has size at least $\log \log n$ is $1/\log^c n$ for sufficiently small a. Then for each edge, we know that removing it causes each of its two incident vertices to have component size less than $\log \log n - 1$ with $1 - 1/\log^c n$ probability. Since the rank is one more than the size of one of these components, the rank is less than $\log \log n$ with $1 - 2/\log^c n$ probability.

Therefore, the expected number of edges with rank at least $\log \log n$ is $2k/\log^c n$. Hence, with probability $1 - \alpha/2$ there are at most $(1/\alpha) 4k/\log^c n$ such edges; adjusting c gives the result. ∎

Lemma 5.4: Let ROBUST2DSFFT' be a modified ROBUST2DSFFT that avoids false negatives or bad updates: whenever a false negative or bad update would occur, an oracle corrects the algorithm. With large constant probability, ROBUST2DSFFT' recovers $\widehat{z}$ such that there

exists a $(k/\log^c n)$-sparse $\widehat{z}'$ satisfying

$$\|\widehat{z} - \widehat{x} - \widehat{z}'\|_2^2 \leq 6\sigma^2 n.$$

Furthermore, only $O(k/\log^c n)$ false negtives or bad updates are caught by the oracle.

Proof: One can choose the random $\widehat{x^*}$ by first selecting the topology of the graph G, and then selecting the random ordering of the columns and rows of the matrix. Note that reordering the vertices only affects the ideal ordering by a permutation within each stage of recovery; the set of edges recovered at each stage in the ideal ordering depends only on the topology of G. Suppose that the choice of the topology of the graph satisfies the thesis of Lemma 5.3 (which occurs with large constant probability). We will show that with large constant probability (over the space of random permutations of the rows and columns), ROBUST2DSFFT' follows the ideal ordering and the requirements of Lemma 5.1 are satisfied at every stage.

For a recovered edge e, we define the "residue" $\widehat{x^*}_e - \widehat{z}_e$. We will show that if e has rank r, then $|\widehat{x^*}_e - \widehat{z}_e| \leq r\sqrt{\frac{\log\log n}{\log n}}\epsilon L$.

During attempted recovery at any vertex v during the ideal ordering (including attempts on vertices which do not have exactly one incident edge), let $y \in \mathbb{C}^{\sqrt{n}}$ be the associated column/row of $\widehat{x} - \widehat{z}$. We split y into three parts $y = y^{head} + y^{residue} + y^{gauss}$, where y^{head} contains the elements of $\widehat{x^*}$ not in $\mathrm{supp}(\widehat{z})$, $y^{residue}$ contains $\widehat{x^*} - \widehat{z}$ over the support of $\widehat{z}$, and y^{gauss} contains $\widehat{w}$ (all restricted to the column/row corresponding to v). Let $S = \mathrm{supp}(y^{residue})$ contain the set of edges incident on v that have been recovered so far. We have by the inductive hypothesis that $\|y^{residue}\|_1 \leq \sum_{e\in S} \mathrm{rank}(e)\sqrt{\frac{\log\log n}{\log n}}\epsilon L$. Since the algorithm verifies that $\sum_{e\in S}\mathrm{rank}(e) \leq \log\log n$, we have

$$\|y^{residue}\|_1 \leq \sqrt{\frac{\log^3\log n}{\log n}}\epsilon L < \epsilon L.$$

Furthermore, y is permutation invariant: if we condition on the values and permute the rows and columns of the matrix, the algorithm will consider the permuted y in the same stage of the algorithm.

Therefore, the conditions for Lemma 5.1 hold. This means that the chance of a false positive is $1/n^c$, so by a union bound, this never occurs. Because false negatives never occur by assumption, this means we continue following the ideal ordering. Because bad updates never occur, new residuals have magnitude at most

$$\|y^{residue}\|_1 + \sqrt{\frac{\log\log n}{\log n}}\epsilon L.$$

Because $\|y^{residue}\|_1 / \left(\sqrt{\frac{\log\log n}{\log n}}\epsilon L\right) \leq \sum_{e\in S}\mathrm{rank}(e) = \mathrm{rank}(v) = \mathrm{rank}(e) - 1$, each new residual has magnitude at most

$$\mathrm{rank}(e)\sqrt{\frac{\log\log n}{\log n}}\epsilon L \leq \epsilon L. \tag{2}$$

as needed to complete the induction.

Given that we follow the ideal ordering, we recover every edge of rank at most $\log\log n$. Furthermore, the residue on every edge we recover is at most ϵL. By Lemma 5.3, there are at most $k/\log^c n$ edges that we do not recover. From Equation (2), the squared ℓ_2 norm of the residues is at most $\epsilon^2 L^2 k = \epsilon^2 C^2\sigma^2 n/k\cdot k < \sigma^2 n$ for ϵ small enough. Since $\|\widehat{w}\|_2^2 < 2\sigma^2 n$ with overwhelming probability, there exists a $\widehat{z}'$ so that

$$\|\widehat{z} - \widehat{x} - \widehat{z}'\|_2^2 \leq 2\|\widehat{z} - \widehat{x^*} - \widehat{z}'\|_2^2 + 2\|w\|_2^2 \leq 6\sigma^2 n.$$

Finally, we need to bound the number of times the oracle catches false negatives or bad updates. The algorithm applies Lemma 5.1 only $2\sqrt{n}+O(k) = O(k)$ times. Each time has a $1/\log^c n$ chance of a false negatives or bad update. Hence the expected number of false negatives or bad updates is $O(k/\log^c n)$. ∎

Lemma 5.5: For any constant $\alpha > 0$, the algorithm ROBUST2DSFFT can with probability $1-\alpha$ recover $\widehat{z}$ such that there exists a $(k/\log^{c-1} n)$-sparse $\widehat{z}'$ satisfying

$$\|\widehat{z} - \widehat{x} - \widehat{z}'\|_2^2 \leq 6\sigma^2 n$$

using $O(k\log n)$ samples and $O(k\log^2 n)$ time.

Proof: To do this, we will show that changing the effect of a single call to ROBUSTESTIMATECOL can only affect $\log n$ positions in the output of ROBUST2DSFFT. By Lemma 5.4, we can, with large constant probability, turn ROBUST2DSFFT into ROBUST2DSFFT' with only $O(k/\log^c n)$ changes to calls to ROBUSTESTIMATECOL. This means the outputs of ROBUST2DSFFT and of ROBUST2DSFFT' only differ in $O(k/\log^{c-1} n)$ positions.

We view ROBUSTESTIMATECOL as trying to estimate a vertex. Modifying it can change from recovering one edge (or none) to recovering a different edge (or none). Thus, a change can only affect at most two calls to ROBUSTESTIMATECOL in the next stage. Hence in r stages, at most 2^{r-1} calls may be affected, so at most 2^r edges may be recovered differently.

Because we refuse to recover any edge with rank at least $\log\log n$, the algorithm has at most $\log\log n$ stages. Hence at most $\log n$ edges may be recovered differently as a result of a single change to ROBUSTESTIMATECOL. ∎

The algorithm in [14] can be generalized to the 2 dimensional case. The generalization can be found in [9].

Here, we restate the theorem which we will use to prove the correctness of our ROBUST2DSFFT algorithm.

Theorem 5.6: There is a variant of [14] algorithm that will, given $x, \widehat{z} \in \mathbb{C}^{\sqrt{n}\times\sqrt{n}}$, return $\widehat{x'}$ with

$$\|\widehat{x} - \widehat{z} - \widehat{x'}\|_2 \leq 2 \cdot \min_{k\text{-sparse } \widehat{x^*}} \|\widehat{x} - \widehat{z} - \widehat{x^*}\|_2^2 + \|\widehat{x}\|_2^2/n^c$$

with probability $1 - \alpha$ for any constants $c, \alpha > 0$ in time

$$O(k \log(n/k) \log^2 n + |\mathrm{supp}(\widehat{z})| \log(n/k) \log n),$$

using $O(k \log(n/k) \log^2 n)$ samples of x.

Theorem 5.7: Our overall algorithm can recover $\widehat{x'}$ satisfying

$$\|\widehat{x} - \widehat{x}'\|_2^2 \leq 12\sigma^2 n + \|\widehat{x}\|_2^2/n^c$$

with probability $1 - \alpha$ for any constants $c, \alpha > 0$ in $O(k \log n)$ samples and $O(k \log^2 n)$ time, where $k = a\sqrt{n}$ for some constant $a > 0$.

Proof: By Lemma 5.5, we can recover an $O(k)$-sparse $\widehat{z}$ such that there exists an $(k/\log^{c-1} n)$-sparse $\widehat{z'}$ with

$$\|\widehat{x} - \widehat{z} - \widehat{z}'\|_2^2 \leq 6\sigma^2 n.$$

with arbitrarily large constant probability for any constant c using $O(k \log^2 n)$ time and $O(k \log n)$ samples. Then by Theorem 5.6, we can recover a $\widehat{z'}$ in $O(k \log^2 n)$ time and $O(k \log^{4-c} n)$ samples satisfying

$$\|\widehat{x} - \widehat{z} - \widehat{z'}\|_2^2 \leq 12\sigma^2 n + \|\widehat{x}\|_2^2/n^c$$

and hence $\widehat{x'} := \widehat{z} + \widehat{z'}$ is a good reconstruction for $\widehat{x}$. ■

REFERENCES

[1] A. Akavia. Deterministic sparse Fourier approximation via fooling arithmetic progressions. *COLT*, pages 381–393, 2010.

[2] A. Akavia, S. Goldwasser, and S. Safra. Proving hard-core predicates using list decoding. *FOCS*, 44:146–159, 2003.

[3] M. Akcakaya and V. Tarokh. A frame construction and a universal distortion bound for sparse representations. *Signal Processing, IEEE Transactions on*, 56(6):2443 –2450, june 2008.

[4] O. C. Andronesi, G. S. Kim, E. Gerstner, T. Batchelor, A. A. Tzika, V. R. Fantin, M. G. V. Heiden, and A. G. Sorensen. Detection of 2-hydroxyglutarate in idh-mutated glioma patients by in vivo spectral-editing and 2d correlation magnetic resonance spectroscopy. 4:116ra4, 2012.

[5] V. Bahskarna and K. Konstantinides. *Image and video compression standards : algorithms and architectures*. Kluwer Academic Publishers, 1995.

[6] P. Boufounos, V. Cevher, A. C. Gilbert, Y. Li, and M. J. Strauss. What's the frequency, kenneth?: Sublinear fourier sampling off the grid. *RANDOM/APPROX*, 2012.

[7] E. Candes and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies. *IEEE Trans. on Info.Theory*, 2006.

[8] Y. Chan and V. Koo. An Introduction to Synthetic Aperture Radar (SAR). *Progress In Electromagnetics Research B*, 2008.

[9] B. Ghazi, H. Hassanieh, P. Indyk, D. Katabi, E. Price, and L. Shi. Sample-Optimal Average-Case Sparse Fourier Transform in Two Dimensions. *arXiv:1303.1209*, 2013.

[10] A. Gilbert, S. Guha, P. Indyk, M. Muthukrishnan, and M. Strauss. Near-optimal sparse Fourier representations via sampling. *STOC*, 2002.

[11] A. Gilbert, M. Muthukrishnan, and M. Strauss. Improved time bounds for near-optimal space Fourier representations. *SPIE Conference, Wavelets*, 2005.

[12] B. G. Haskell, A. Puri, and A. N. Netravali. *Digital video : an introduction to MPEG-2*. Chapman and Hall, 1997.

[13] H. Hassanieh, F. Adib, D. Katabi, and P. Indyk. Faster gps via the sparse fourier transform. *MOBICOM*, 2012.

[14] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Near-optimal algorithm for sparse Fourier transform. *STOC*, 2012.

[15] H. Hassanieh, P. Indyk, D. Katabi, and E. Price. Simple and practical algorithm for sparse Fourier transform. *SODA*, 2012.

[16] M. Iwen. Improved approximation guarantees for sublinear-time Fourier algorithms. *Applied And Computational Harmonic Analysis*, 2012.

[17] M. A. Iwen. Combinatorial sublinear-time Fourier algorithms. *Foundations of Computational Mathematics*, 10:303–338, 2010.

[18] M. A. Iwen, A. Gilbert, and M. Strauss. Empirical evaluation of a sub-linear time sparse dft algorithm. *Communications in Mathematical Sciences*, 5, 2007.

[19] A. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging*. Society for Industrial and Applied Mathematics, 2001.

[20] K. Kazimierczuk and V. YU. Accelerated nmr spectroscopy by using compressed sensing. *Angewandte Chemie International Edition*, 2011.

[21] D. Lawlor, Y. Wang, and A. Christlieb. Adaptive sub-linear time fourier algorithms. *arXiv:1207.6368*, 2012.

[22] M. Lustig, D. Donoho, J. Santos, and J. Pauly. Compressed sensing mri. *Signal Processing Magazine, IEEE*, 25(2):72–82, 2008.

[23] Y. Mansour. Randomized interpolation and approximation of sparse polynomials. *ICALP*, 1992.

[24] D. Nishimura. *Principles of Magnetic Resonance Imaging*. Society for Industrial and, 2010.

[25] S. Pawar and K. Ramchandran. Computing a k-sparse n-length Discrete Fourier Transform using at most 4k samples and O(k log k) complexity . In *ISIT*, 2013.

[26] E. Price and D. P. Woodruff. $(1+\epsilon)$-approximate sparse recovery. *FOCS*, 2011.

[27] E. E. Schadt, M. D. Linderman, J. Sorenson, L. Lee, and G. P. Nolan. Computational solutions to large-scale data management and analysis. 2011.

[28] L. Shi, O. Andronesi, H. Hassanieh, B. Ghazi, D. Katabi, and E. Adalsteinsson. MRS Sparse-FFT: Reducing Acquisition Time and Artifacts for In Vivo 2D Correlation Spectroscopy. In *ISMRM'13, Int. Society for Magnetic Resonance in Medicine Annual Meeting and Exhibition*, 2013.

[29] E. Sidky. What does compressive sensing mean for X-ray CT and comparisons with its MRI application. In *Conference on Mathematics of Medical Imaging*, 2011.

[30] G. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 1991.

[31] O. Yilmaz. *Seismic Data Analysis: Processing, Inversion, and Interpretation of Seismic Data*. Society of Exploration Geophysicists, 2008.

[32] J. Yoo, S. Becker, M. Loh, M. Monge, E. Candès, and A. E-Neyestanak. A 100MHz–2GHz 12.5x subNyquist rate receiver in 90nm CMOS. In *IEEE RFIC*, 2012.