3-D Closed-Form Sparse Reconstruction from Downsampled Discrete Fourier Transform Data

Andrew E. Yagle
Department of EECS, The University of Michigan, Ann Arbor, MI 48109-2122

Abstract— **We present a simple closed-form algorithm for reconstructing a sparse 3-D image from three different sets of downsampled discrete Fourier transform (DFT) values. Unlike our previous work on this problem, we do not use an indicator function or POCS. Instead, we compare the inverse DFTs of the three sets of downsampled DFT values at each point. Where two of the three values agree, the image value is determined. We then bootstrap the problem by subtracting off the DFT of these known image values, producing a problem like the original but with a much sparser image. Repeating this two more times produces either a zero image, or a problem so sparse that it can be solved by solving a very small linear system. A $(60 \times 60 \times 60)$ 624-sparse example is given.**

Keywords— **Sparse reconstruction**
Phone: 734-763-9810. Fax: 734-763-1503.
Email: aey@eecs.umich.edu.

I. INTRODUCTION

A. Problem Statement

The $(N \times N \times N)$-point 3-D DFT X_{k_1,k_2,k_3} of the $(N \times N \times N)$ 3-D image x_{n_1,n_2,n_3} is

$$X_{k_1,k_2,k_3} = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1}\sum_{n_3=0}^{N-1} x_{n_1,n_2,n_3} e^{-j\frac{2\pi}{N}(n_1k_1+n_2k_2+n_3k_3)} \tag{1}$$

where $0 \leq k_1, k_2, k_3 \leq N–1$. x_{n_1,n_2,n_3} is mostly zero-valued, and the locations of its nonzero values are unknown. x_{n_1,n_2,n_3} need *not* be real-valued; in fact, the algorithm works better for complex x_{n_1,n_2,n_3}.

The 3-D DFT X_{k_1,k_2,k_3} is known *only* for the following three sets of values of $\{k_1, k_2, k_3\}$:

- $k_i \in \{0, L_1, 2L_1, 3L_1 \ldots N - L_1\}, i = 1, 2, 3.$
- $k_i \in \{0, L_2, 2L_2, 3L_2 \ldots N - L_2\}, i = 1, 2, 3.$
- $k_i \in \{0, L_3, 2L_3, 3L_3 \ldots N - L_3\}, i = 1, 2, 3.$
- L_1, L_2, L_3 are all integral factors of N.
- L_1, L_2, L_3 are pairwise relatively prime.

The goal is to recover the sparse image x_{n_1,n_2,n_3} from these known DFT values. Note half of the given DFT values are complex conjugates of the other half.

B. Other Approaches

A common approach to sparse reconstruction is to compute the minimum ℓ_1 norm solution, perhaps by linear programming. If the DFT wavenumbers are randomly chosen, and if enough of them are known, then it has been shown that the minimum ℓ_1 norm solution is in fact x_{n_1,n_2,n_3}. Other approaches include thresholded Landweber iteration and orthogonal matching pursuit. Since our apporach is completely different from all of these, we refer the reader to the extensive literature on all of these methods.

II. Presentation of Algorithm

A. Replace Unknown Values of X_{k_1,k_2,k_3} with Zeros

Given the above data, define

$$Y_{k_1,k_2,k_3} = \begin{cases} X_{k_1,k_2,k_3} & \text{for } k_i{=}0, L, 2L \ldots N{-}L \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The inverse $(N \times N \times N)$-point DFT of Y_{k_1,k_2,k_3} is

$$y_{n_1,n_2,n_3} = \frac{1}{L^3}\sum_{i_1=0}^{L-1}\sum_{i_2=0}^{L-1}\sum_{i_3=0}^{L-1} x_{n_1+i_1\frac{N}{L},n_2+i_2\frac{N}{L},n_3+i_3\frac{N}{L}}. \tag{3}$$

So inserting zeros for the missing data and computing the inverse $(N \times N \times N)$-point DFT gives L^3 copies of the nonzero values of x_{n_1,n_2,n_3}, each copy shifted in each of the n_i by an integer multiple of N/L.

Applying this to each of the three sets of data gives

$$\begin{aligned} y^{(1)}_{n_1,n_2,n_3} &= \frac{1}{L_1^3}\sum_{i_1=0}^{L_1-1}\sum_{i_2=0}^{L_1-1}\sum_{i_3=0}^{L_1-1} x_{n_1+i_1\frac{N}{L_1},n_2+i_2\frac{N}{L_1},n_3+i_3\frac{N}{L_1}}. \\ y^{(2)}_{n_1,n_2,n_3} &= \frac{1}{L_2^3}\sum_{i_1=0}^{L_2-1}\sum_{i_2=0}^{L_2-1}\sum_{i_3=0}^{L_2-1} x_{n_1+i_1\frac{N}{L_2},n_2+i_2\frac{N}{L_2},n_3+i_3\frac{N}{L_2}}. \\ y^{(3)}_{n_1,n_2,n_3} &= \frac{1}{L_3^3}\sum_{i_1=0}^{L_3-1}\sum_{i_2=0}^{L_3-1}\sum_{i_3=0}^{L_3-1} x_{n_1+i_1\frac{N}{L_3},n_2+i_2\frac{N}{L_3},n_3+i_3\frac{N}{L_3}}. \end{aligned} \tag{4}$$

B. Previous Algorithm: Indicator Function

Previously, we used the product

$$y_{n_1,n_2,n_3} = y^{(1)}_{n_1,n_2,n_3} y^{(2)}_{n_1,n_2,n_3} y^{(3)}_{n_1,n_2,n_3} \tag{5}$$

as an indicator function for locations of nonzero values of x_{n_1,n_2,n_3}. y_{n_1,n_2,n_3} is nonzero only if all three $y^i_{n_1,n_2,n_3}$ are nonzero, and this indicates a possible location of a nonzero value of x_{n_1,n_2,n_3}. However, there are many false alarms, where all three $y^i_{n_1,n_2,n_3}$ are aliases of x_{n_1,n_2,n_3}, unless x_{n_1,n_2,n_3} is very sparse.

C. *New Algorithm: Voting*

Instead, we use the following procedure:

- For each location $\{n_1, n_2, n_3\}, 0 \leq n_1, n_2, n_3 < N$:
- If *any two* of the three values of $y^{(i)}_{n_1,n_2,n_3}$ agree, then declare that x_{n_1,n_2,n_3} is that agreed value;
- Otherwise, x_{n_1,n_2,n_3} is declared unknown.

Next, *bootstrap* the problem as follows:

- Compute the 3-D DFT of the values of x_{n_1,n_2,n_3} declared by this procedure, with zeros elsewhere.
- Subtract these 3-D DFT values from the data.
- This produces another problem like the original, but with a much sparser image than the original, since the declared values have been removed.

Bootstraping this problem in turn produces a problem like the original, but with a *very* sparse image. Bootstraping can be performed again until either:

- No unknown values of x_{n_1,n_2,n_3} are left;
- Few unknown values of x_{n_1,n_2,n_3} are left and these can be computed by solving a very small linear system of equations.

III. Numerical Example

We reconstruct a 624-sparse $(60 \times 60 \times 60)$ 3-D image from 3 sets of downsampled 3-D DFT values with L_1=3;L_2=4;L_3=5. The number of observations is

$$M = \left(\frac{60}{3}\right)^3 + \left(\frac{60}{4}\right)^3 + \left(\frac{60}{5}\right)^3 = 13103. \quad (6)$$

- This includes conjugate symmetric values.
- It also counts some DC values multiple times.
- $\frac{1}{2}\frac{13103}{60^3}$=3% of all of the DFT values are known.

Each stage of the algorithm reduced the sparsity:

- 1^{st} stage: 624→46. 2^{nd}: 46→3. 3^{rd}: 3→0.

The third stage could have been replaced by solving a 3×3 linear system of equations for the still-unknown values of the 3-D image x_{n_1,n_2,n_3}.

The original and reconstructed 3-D images coincide, so only one figure is shown. The colors indicate the values of the 3-D image x_{n_1,n_2,n_3}.

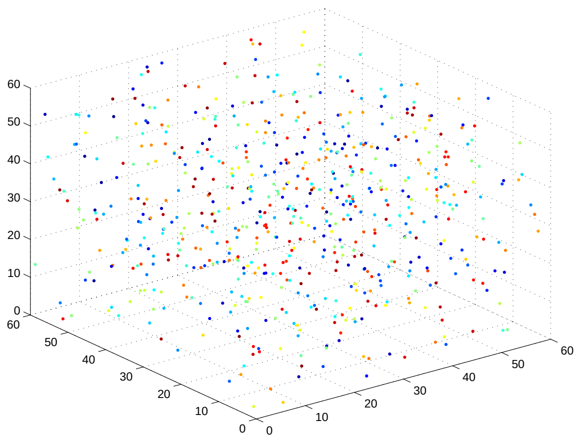

Fig. 1. (60×60×60) 624-sparse 3-D image. The original and reconstructed images coincide. Color denote image values.

A. *Matlab Program*

```
clear;N1=3;N2=4;N3=5;N=N1*N2*N3;
T=0.997;X=rand(N,N,N);X(X<T)=0;
X=X.*(rand(N,N,N)+1);%X random values
FX=fftn(X);%Get X from downsampled FX:
F1=FX(1:N1:N,1:N1:N,1:N1:N);%Downsample N1
F2=FX(1:N2:N,1:N2:N,1:N2:N);%Downsample N2
F3=FX(1:N3:N,1:N3:N,1:N3:N);%Downsample N3
%GOAL: X from F1,F2,F3.  SOLUTION:
F1H(N/N1,N/N1,N/N1)=0;G1(N,N,N)=0;
F2H(N/N2,N/N2,N/N2)=0;G2(N,N,N)=0;
F3H(N/N3,N/N3,N/N3)=0;G3(N,N,N)=0;
XH(N,N,N)=0;XHAT(N,N,N)=0;
for I=1:3;%#of bootstraps
G1(1:N1:N,1:N1:N,1:N1:N)=F1-F1H;
G2(1:N2:N,1:N2:N,1:N2:N)=F2-F2H;
G3(1:N3:N,1:N3:N,1:N3:N)=F3-F3H;
X1=N1*N1*N1*real(ifftn(G1));
X2=N2*N2*N2*real(ifftn(G2));
X3=N3*N3*N3*real(ifftn(G3));
%If any 2 agree, use them for XH:
XH(abs(X1-X2)<0.00001)=X1(abs(X1-X2)<0.00001);
XH(abs(X1-X3)<0.00001)=X1(abs(X1-X3)<0.00001);
XH(abs(X2-X3)<0.00001)=X2(abs(X2-X3)<0.00001);
XHAT=XHAT+XH;FXHAT=fftn(XHAT);%BOOTSTRAP:
F1H=FXHAT(1:N1:N,1:N1:N,1:N1:N);
F2H=FXHAT(1:N2:N,1:N2:N,1:N2:N);
F3H=FXHAT(1:N3:N,1:N3:N,1:N3:N);end;
%"ind2sub" maps 1D index from "find" to 3D
[IX,JX,KX]=ind2sub([N,N,N],find(abs(X)>.001));
[IY,JY,KY]=ind2sub([N,N,N],find(abs(XHAT)>.001));
figure,title('ORIGINAL IMAGE; COLOR IS VALUE'),
scatter3(IX,JX,KX,5,X(abs(X)>0.001),'filled')
figure,title('COMPUTED IMAGE; COLOR IS VALUE'),
scatter3(IY,JY,KY,5,X(abs(XHAT)>0.001),'filled')
```