Accelerating nonuniform fast Fourier transform via reduction in memory access latency

Nihshanka Debroy[a], Nikos P. Pitsianis[ab] and Xiaobai Sun[a]

[a] Department of Computer Science, Duke University, Durham, NC 27708, USA
[b] Department of Electrical and Computer Engineering, Aristotle University, Thessaloniki, 54124, Greece

ABSTRACT

We address the discrepancy that existed between the low arithmetic complexity of nonuniform Fast Fourier Transform (NUFFT) algorithms and high latency in practical use of NUFFTs with large data sets, especially, in multi-dimensional domains. The execution time of a NUFFT can be longer by a factor of two orders of magnitude than what is expected by the arithmetic complexity. We examine the architectural factors in the latency, primarily on the non-even latency distribution in memory references across different levels in the memory hierarchy. We then introduce an effective approach to reducing the latency substantially by exploiting the geometric features in the sample translation stage and making memory references local. The restructured NUFFT algorithms render efficient computation in sequential as well as in parallel. Experimental results and improvements for radially encoded magnetic resonance image reconstruction are presented.

Keyword list: FFT, NUFFT, memory latency, image reconstruction

1. INTRODUCTION

The term USFFT, or NUFFT, stands for fast Fourier transforms with data located at unequally spaced Cartesian coordinates, or in non-uniform sampling density. In fact there are two sets of data in a discrete Fourier transform (DFT), a set of source data at the input and a set of target data as the outcome of numerical evaluation. The celebrated fast Fourier transform (FFT) by Cooley and Tukey[5] and its numerous variants assume that both the source and target data are located at equally spaced Cartesian grid points in their respective domains. In many signal and image processing applications, especially in multi-dimensional source and target domains, the data are not sampled equally spaced on a Cartesian grid, see[13] for instance. To utilize the FFT, various interpolation techniques had been used ad hoc or heuristically, in dealing with the difference between a non-equally spaced sampling scheme and the FFT condition on data location. In comparison to the precursor efforts, NUFFT algorithms provide systematic control on the gain or loss in numerical accuracy, accounting for the sample deviation from the FFT condition, while maintaining the amount of arithmetic operations within the same order as that of the FFT. Such algorithms have emerged since the seminal paper by Dutt and Rokhlin[6] and gained broader recognition in their applications. Among other algorithm alternatives are those by Beylkin,[4] Liu and Nguyen[11,12] and Greengard and Lee.[9] NUFFT methods specific to data translation from a polar or spherical grid to a Cartesian one include the Kaiser-Bessel gridding method, see[3,10] and the references therein.

In this paper we are concerned with the substantial impact of architectural features on the latency in computing NUFFTs with large data sets, which in some cases include several hundred million sample points or more. In the study of small animal organs with magnetic resonance (MR) microscopy imaging, for instance, a three-dimensional image formation involves large sample ensembles. In a particular setting, the data are acquired in the so-called k space, following the radial encoding/scanning scheme shown in Figure 1. The sample locations are neither equally spaced in Cartesian coordinates nor uniformly distributed in sample density. The size of the sample ensembles is up to 421 million datum points at present and is expected to reach 3.3 billion points in the near future. The direct evaluation of the discrete Fourier transform with non-equally spaced data is not a feasible option. Yet, the execution time of a NUFFT was far from expected. In a particular experiment, a NUFFT with

Send correspondence to Nikos.P.Pitsianis@Duke.edu

Advanced Signal Processing Algorithms, Architectures, and Implementations XVIII,
edited by Franklin T. Luk, Proc. of SPIE Vol. 7074, 707404, (2008) · 0277-786X/08/$18 · doi: 10.1117/12.795208

421 million samples took 27 hours, slower by a factor of two orders of magnitude than what is expected by the arithmetic complexity and the processing speed of modern processors in use.

We identified the large and undesirable discrepancy in latency as the result of memory consumption and contention at different memory levels and non-even latency distribution in memory accesses across different levels of the memory hierarchy. We introduce a simple model for describing the complexity of memory access latency, besides the arithmetic complexity, and we present an approach to reducing the latency. In this approach, we exploit the geometric features in the sample translation stage of a NUFFT algorithm, improve the locality in memory references with respect to certain constraints in data placement, and restructure the algorithms so that the memory consumption at each algorithm step is reduced and the memory contention is eliminated. The approach is effectively applied to and demonstrated with a particular application of the NUFFT in radially encoded MR imaging.

The rest of the paper is organized as follows. In Section 2 we describe a family of the NUFFT algorithms that utilize the FFT via a sample translation scheme. In Section 3 we describe first the architectural impact on the latency in computing NUFFTs and then give a diagnostic analysis. In Section 4 we introduce an acceleration approach and present experimental results. We conclude the paper in Section 5 with additional discussion on certain remaining problems and potential extensions of the acceleration approach.

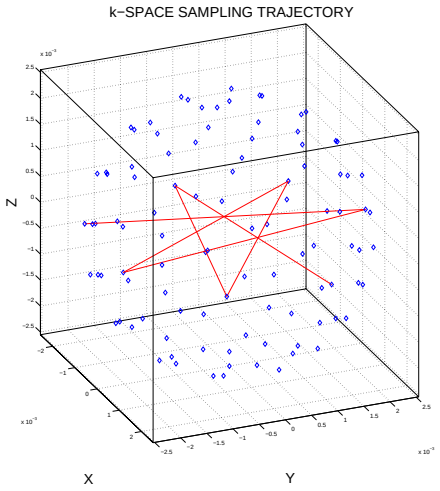

Figure 1. A radial encoding/scanning scheme: the samples are nearly equally-spaced on each ray from the origin to an end point on a spherical surface. The scanning trajectory changes 157 degrees between two consecutive rays. The red line connecting the sample points indicates the acquisition sequence.

2. ALGORITHM DESCRIPTION

We describe a family of the NUFFT algorithms that utilize the FFT via a sample translation scheme. In the case of image formation from spectral data at non-equally spaced Cartesian coordinates, a NUFFT algorithm in this family first translates the sample data, with proper density compensation coefficients, onto an equally-spaced Cartesian grid by convolution with a chosen kernel function, then applies the inverse FFT, and obtains with a final scaling an image formation in an image field of view on an equally spaced pixel or voxel grid. This may be formally expressed in an approximate matrix factorization form. Denote by $\mathbf{u}(S)$ the spectral data sampled at point set S and denote by $\mathbf{v}(T)$ the transformed data at target sample set T. The computation of $\mathbf{v}(T)$ by a NUFFT algorithm may be described as follows,

$$\mathbf{d}(T) \odot (\mathbf{v}(T) + \mathbf{e}(T)) = \mathbf{F}^{-1}(T, \tilde{S}) \cdot \mathbf{C}(\tilde{S}, S) \cdot \mathbf{u}(S), \quad \max_{t \in T} |\epsilon(t)| < \epsilon_b, \tag{1}$$

where $\mathbf{C}$ is the convolution matrix defined by a real-valued convolution kernel function $\mathbf{c}(t-s)$ over $S \times \bar{S}$, $\bar{S}$ is the set of re-sample locations at equally spaced Cartesian grid points in the source domain. It is determined

that $|\bar{S}| = \alpha|S| = |T|$ with $\alpha > 1$ as the oversampling factor, which is often less than 2. The sample translation by convolution is followed by the inverse FFT with the matrix $\mathbf{F}^{-1}(T, \bar{S})$, transforming the spectral data at $\tilde{S}$ to the image domain. The image formation is obtained by scaling the transformed data point-wise with function $\mathbf{d}(t)$, which is the Fourier transform of the convolution kernel function. The approximation error $\mathbf{e}$ in $\mathbf{v}$ can be made below a prescribed error bound $\epsilon_b > 0$ by proper choice of the convolution-scaling function pair and the oversampling factor. The convolution kernel function is chosen so that its support is as local as possible and its Fourier partner used as the scaling function in the image space does not have zeros in the finite scaling domain.

The NUFFT algorithms in the family described in (1) differ primarily in the way the convolution and scaling functions are chosen, determined or evaluated. In the first NUFFT algorithm by Dutt and Rokhlin,[6] the convolution-scaling functions are Gaussian. Specifically, when the convolution kernel function is $c(s) = e^{-bs^2}$, the corresponding scaling function in the image domain is $d(t) = \sqrt{\pi/b} \cdot e^{\frac{-\pi^2 t^2}{b}}$, where b is the parameter that is related to the numerical support of the convolution and determined by the accuracy requirement. In Beylkin's algorithm,[4] $c(s)$ is a central B-spline of odd order m, which has analytically finite support, and $d(t) = (\sin(\pi t)/(\pi t))^{m+1}$. Here we omit the scaling in variables s and t. In the algorithm by Liu and Nguyen,[11,12] the cosine function is used as the scaling function, $d(t) = \cos(\pi t/(\alpha L))$, where α is the oversampling factor and L is the side length of the Cartesian grid in the image field of view. The convolution coefficients are then determined numerically by a local least-squares interpolation of the discrete non-equally-spaced Fourier transform matrix, based on the local support radius suggested by the Dutt-Rokhlin algorithm. The min-max criterion is used as an alternative in the algorithm by Fessler and Sutton.[7] The numerically determined convolution coefficients are specific to sample locations. In each of the NUFFT algorithms, the translation-scaling coefficients can be either pre-computed and saved or evaluated on the fly as they are needed.

Consider now the arithmetic complexity of a NUFFT algorithm, namely, the number of arithmetic operations the algorithm requires. Let $N = |S|$ be the size of the spectral sample ensemble. Then $|\tilde{S}| = |T| = \alpha N$ is the number of re-sampled data points in the spectral domain, which equals the number of data points in the image domain. The sample translation by convolution takes $4rN$ arithmetic operations with complex data, where r is the number of re-sample locations within the local support of the convolution at each original sample location. The translation parameter r increases with the radius of the convolution support, with the geometric dimension of the source domain, and increases reciprocally with the decrease in ϵ_b. The inverse FFT takes no more than $5|T|\log(|T|)$ operations. Finally, the scaling in the image domain requires only $|T|$ scalar multiplications or divisions. In total, a NUFFT algorithm takes $O(\alpha N \log(\alpha N) + r(d, \epsilon_b^{-1})N + N)$ arithmetic operations, when the convolution and scaling coefficients are precomputed. The arithmetic complexity is only a couple of times higher when the convolution and scaling coefficients are evaluated on the fly. Table 1 illustrates the comparison in time (in Matlab) between two NUFFT algorithms with $r = 5^3$ and the naive evaluation of $\mathbf{v}(T)$. Evidently, the direct method is too slow, even on a small 3D grid.

Table 1. Comparison in execution time (minutes), with two different grid sizes, between the direct evaluation of the DFT and two NUFFT algorithms based on Liu-Nguyen (NUFFT-LN) method and Dutt-Rokhlin (NUFFT-DR) method respectively.

Grid Size	Direct DFT	NUFFT-LN	NUFFT-DR
$32 \times 32 \times 32$	16	.283	.233
$64 \times 64 \times 64$	1920	2.470	1.970

We add that while it is well known that an equally spaced convolution with broad support can be accelerated via the use of the FFT, based on the discrete convolution theorem, NUFFT algorithms exploit the local-support convolution to accelerate the calculation of non-equally spaced DFT by (1), which states a discrete approximate convolution theorem.

3. ARCHITECTURAL LATENCY ANALYSIS

In this section we address the architectural issues that affect substantially the latency of NUFFT in execution, besides the arithmetic complexity, and dominate the execution time when the data sets are large. Table 2 manifests the architectural impact on the latency of NUFFT in execution. In both the FFT and NUFFT, a

Cartesian grid of size $768 \times 768 \times 768$ is used. The FFT time is by the use of the FFTW library[8] in a stand-alone setup (without multi-threading), with complex-valued data. The NUFFT time accounts for that in both sample translation and the FFT stages. The translation locality parameter $r = 5^3$, i.e., every source datum value at a point in S affects 125 neighbor points in $\tilde{S}$. The huge discrepancy in the execution time on the machine with smaller memory capacity is far beyond what is expected by the arithmetic complexity and the processing speed. In the experiment, the sample coordinates and datum values are provided to the NUFFT algorithm in large arrays.

Table 2. Execution time (in minutes) of the stand-alone FFT and NUFFT with a Cartesian grid of size $768 \times 768 \times 768$.

Machine	FFT	NUFFT
8 GB RAM, 8-core, 3.0 GHz	2.2	720
12 GB RAM, 4-core, 2.5 GHz	12.0	240

3.1 Architectural latency factors

We describe two primary architectural latency factors. Consider first the latency due to insufficient memory capacity for large data sets. In particular, we describe the amount of memory space used by the NUFFT algorithm with a particular set of k-space data, provided by the research group at the Center for In Vivo Microscopy, Duke University, for the study of small animal organs such as mouse brains and hearts with MR microscopy imaging. The particular sample ensemble consists of 421 million values acquired by the radial scanning scheme shown in Figure 1. Each sampled data value is a complex number, with real and imaginary parts in double-precision floating point representation. Thus, every sampled value takes 16 bytes in memory. The input data take 6.74 GB in memory. Before sample translation, the row sample ensemble was reduced to 134 million points. Upon over-sampling, the Cartesian grid is of size $768 \times 768 \times 768$, which has about 453 million grid points. The transformed data take 7.25 GB in memory. The memory space required by the sampled data values at the input and transformed data at the output is around 14 GB, exceeding the RAM capacity of the particular machine, without accounting for the memory space by any other input and intermediate data. If the sampling coordinates are to be provided together with the sampled values at the input, additional memory space of 9.2 GB is needed. In this situation, the access to external memory, either by the support of virtual memory system or by the user's out-of-core programming, prolongs the NUFFT execution.

Figure 2 illustrates the impact of external memory accesses, due to excessive memory requirement by NUFFT with large data sets. The data values are complex numbers, with real and complex parts in double precision floating point representation. The experiment is on a modern workstation with a 3.0 GHz processor and 8 giga-bytes(GB) in random access memory (RAM). As the problem size N increases to certain point, the ratio between the execution time and the problem size changes dramatically. For instance, the time to finish the image formation on the $192 \times 192 \times 192$ Cartesian grid is about 1.75 minutes, whereas it takes more than 12 hours on the $768 \times 768 \times 768$ grid. The particular implementation is in practical use, based on the algorithm by Liu and Nguyen ,[11,12] but the architectural impact shown here is typical for the NUFFT algorithm with large data sets.

We describe next the architectural penalty on latency due to the memory references in the memory hierarchy at the end of cache level(s). While the references to the RAM are much faster than those to the external memory, the references at a cache level are much faster than those to the RAM. The cache reference patterns are mainly determined by the data structures and the traversal of every data set in a NUFFT algorithm. Specifically, in the sample translation stage, both the sampled data at the input and the re-sampled data on a Cartesian grid are accessed. Every source datum value at a point in S is used to update re-sampled values at r points in $\tilde{S}$. The data structure for the re-sampled data follows the cube array format commonly assumed by the 3D FFT routines such as that in the FFTW library.[8] In detail, the cube elements are sequentialized along first the z axis, then the y axis, and finally the x axis. The k-space data is in radial format, following the radial scanning scheme shown in Figure 1. In other words, the traversal of the radial data follows each ray from the center to the sphere surface, from one ray to the next, and the end points of two consecutive rays in acquisition are far apart on the sphere surface. As a result of the difference in data placement, the traversal of the radial data along a ray, for example, invokes the zig-zag traversal of the re-sample data in the cube array format, which increases cache misses due to the lack of reference locality.

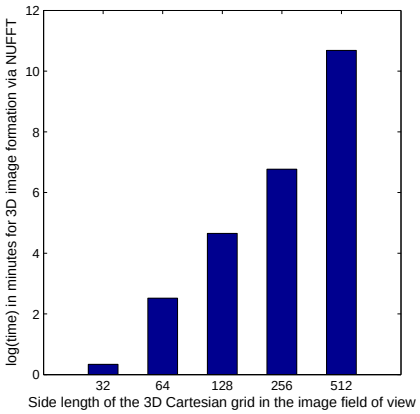

Figure 2. The impact of external memory accesses on the latency of NUFFT in execution on a 3.0 GHz machine with 8 GB RAM.

3.2 Architectural latency complexity model

To formally describe the latency problem in accessing memory at different levels of the memory hierarchy, we make a reference to an I/O model by Aggarwal and Vitter,[2] which is applied in sorting large data,[1] for instance. An I/O operation swaps data between external and internal memory, and it is slower than the access to RAM by a couple of orders in magnitude. Let P be the number of memory units required by a specific problem. Let M be the number of memory units available in internal memory. When $P > M$, there is additional latency caused by accesses to external memory. The I/O model considers the number of times the external memory is accessed. Suppose a block of data on the disk (as the external memory) is swapped at each I/O operation. Let B be the number of memory units per block. In a simple calculation where each memory unit is referenced only once, the number of external memory accesses by disk blocks is $(P - M)/B$ in the best case when each disk block that is retrieved contains B of the $P - M$ units in the external memory. In the worst case when the $P - M$ extra units are in totally different memory blocks, the number of I/O operations is $P - M$.

With the same concepts behind the I/O model, we examine the memory latency in the execution of a NUFFT algorithm. First, the memory requirement increases quickly with $3D$ image formation size. From a modest image formation size and beyond, I/O operations are invoked. Next, on the distribution of the memory units across a data block, we note that as the side length of the cube array increases, the r array elements within the translation distance of a radial datum point in the sample ensemble S become distributed farther apart into more disk blocks, especially, the cube array elements along the x-axis. Furthermore, on the frequency of each memory unit visited, we note that every re-sampled value at a location in $\tilde{S}$ may be updated many times by radial data samples within the sample translation distance defined by the convolution support. Finally, the same analysis applies to the memory references at the cache level, except that the disk block size is replaced by the cache reference size. We have omitted the tedious detail in the complexity model specific to NUFFT computation.

4. LATENCY REDUCTION

By the diagnostic description in Section 3 of the memory latency problem, it is essential to reduce or minimize memory use and make memory references local. To this end, we have taken many steps in re-formatting data structures, arithmetic expressions as well as in improvement of programming structure and techniques. In this paper, we introduce the main step in latency reduction by exploiting the geometric structures in NUFFT algorithms to bring reference locality and parallelism into NUFFTs.

Recall first that the reduction in latency is subject to the constraint in data placement of the cube array as required by the commonly available FFT subroutines, as described in Section 3. We partition the data in k-space. Figure 3 shows a conforming partition for both the radial data on S and the Cartesian data on $\bar{S}$. In each of the octants, the radial data remain intact along each ray. With a pre-specified policy for tie breaking at

the boundary between the octants, every single ray of radial data belongs to one and only one of the octants. The number of rays in each octant is approximately the same as that in another octant.

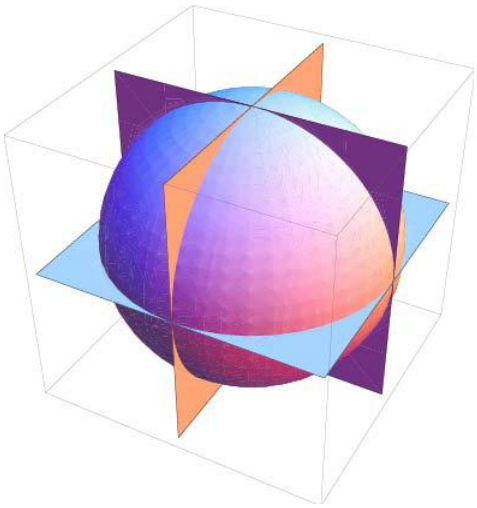

Figure 3. A conforming partition in both the radial data and the Cartesian data

We unpacked and re-grouped the input data according to the partition scheme and divided the sample translation part into 8 stages, where each octal block of the source data is translated in one stage. Thus, the partition scheme introduced memory reference locality into the NUFFT and reduced the memory requirement at each divided translation stage.

To utilize the multi-core architecture, a modification to this octal partition is necessary in order to avoid temporal software fault in potentially concurrent modifications to the same datum. From the source point of view, every single radial sample affects at most r translated samples on the Cartesian grid. From the alternative viewpoint, a translated sample at $\bar{s}_k$ takes or reads contributions from all the source samples within the translation distance. The total number of source samples in its neighborhood is not necessarily known a priori. Due to this asymmetry between the writers and readers in their mutual information, we take the writer's stand and let every source sample update its neighbors in $\bar{S}$ on the Cartesian grid. In this setting, a Cartesian grid point on or close to the axial planes gets updated by its source neighbor points in more than one octant. Its datum value is therefore subject to potential errors in concurrent updating or overwriting.

A mutual exclusion condition is to be set. We establish an exclusion buffer of width w_b between the octants. The source samples in each octant are put into two portions. The first one contains the samples at interior points that are at least $w_b/2$ away from any other octant. The rest belong to the exclusion buffer. In addition, we impose a temporal or scheduling condition that the translation of the samples in an octant's interior portion is not concurrent with that in the buffer. For instance, the translation of the interior samples may be carried out by up to eight threads, without any overlap in their accesses to the Cartesian grid points. This may be followed or preceded by the translation of the buffer samples.

When the total number of samples in the exclusion buffer is much smaller than that in the interior portion of an octant, one may simply use a single thread for the translation of the buffer samples. Otherwise, it may be beneficial to multithread the translation of the buffer samples as well. This entails a further partition in source data and a scheduling rule for mutual exclusion in memory accesses and data updates. In formal description, we categorize the buffer samples according to their degrees in octant interaction. A buffer sample is said to have interaction degree 2 if it is between two and only two interior octants. Geometrically, the source samples of degree 2 occupy 12 plate zones, and each axial plane is associated with 4 of the plates. The source samples of degree 4 fill the pencil gaps between any two degree-2 buffer zones, with 2 pencils along each axis. Finally,

the remaining buffer samples are at or around the origin, with degree 8 in octant interaction. In retrospect, the degree of the interior samples is 1.

The parallelization scheme described above is successfully implemented with 8 concurrent threads, for the translation of degree-1 source samples followed by a single thread for the translation of the samples of higher degrees. We show in Table 3 a progress of the reduction in memory requirement and in total latency for a NUFFT problem with the final Cartesian grid of size $768 \times 768 \times 768$. In each version, a particular latency problem is resolved. The problems resolved in the two early versions are not discussed in this paper. Version 0 is provided by previous work at the Duke Center for In Vivo Microscopy. The convolution and scaling coefficients are evaluated on the fly, with the sample translation locality parameter set as $r = 5^3$. The NUFFT takes 27 hours on the machine with 12 GB memory, 108 times slower than that by version 4. The experiments are based on an implementation in C, compiled using gcc 3.4.6., with option -03 and with 9 threads. We provide the times taken by the sample translation and the FFT separately. One shall notice that the time portion for the FFT step decreases as the memory contention is reduced and finally removed. In the final version, the FFT (parallelized) time is close to that in stand-alone environment. The sum of the translation time and the FFT time is slightly smaller than the total time, which includes the loading of an initial data block and the FFT scheduling.

Table 3. A history of acceleration in execution time (minutes) for a $768 \times 768 \times 768$ image formation : Version 1 - with extraction of common expressions. Version 2 - with consolidation of data arrays and dynamic generation of sample coordinates. Version 3 - with data partition and dynamic data loading. Version 4 - with parallelization using multiple threads.

		12 GB RAM, 4-core, 2.5 GHz			8GB RAM, 8-core, 3.0 GHz		
Version	Memory(GB)	Translation	FFT	Sum	Translation	FFT	Sum
1	30.0	186.0	48.0	234.0	465.0	255.0	720.0
2	14.0	60.0	12.0	72.0	150.0	21.0	171.0
3	7.25	48.0	12.0	60.0	110.0	2.25	112.25
4	7.25	11.25	3.25	14.5	14.0	0.5	14.5

5. CONCLUSION

We have presented effective approaches to introducing locality in memory references and parallelism in NUFFT computation, and narrowed substantially the previously existing gap between low arithmetic complexity of NUFFTs and high latency in practical use of NUFFTs with high-dimensional and large data sets on modern computers. As the memory capacity increases, the desire and demand for image processing with larger data sets increase as well. With the help of such acceleration approaches, NUFFTs are expected to enable computational solutions to more scientific and engineering problems that were computationally intractable.

In the introduced acceleration approach, the latency reduction is subject to the constraint in the placement of the data sets accessed by the FFT. For NUFFTs with even larger data sets, it may be beneficial to lift this constraint in NUFFTs, together with an alternative data placement scheme for large FFTs, especially, in a parallel computing environment.

ACKNOWLEDGMENTS

The authors thank G. Allan Johnson and Sally L. Gewalt for providing the acquired k-space data and insightful discussion on the use of NUFFT algorithms in MR microscopy imaging.

REFERENCES

[1] P. K. Agarwal, L. Arge, T. M. Murali, K. R. Varadarajan, and J. S. Vitter. I/O-efficient algorithms for contour-line extraction and planar graph blocking. *Symposium on Discrete Algorithms (SODA)*, pages 117–126, 1998.

[2] A. Aggarwal and J. S. Vitter. The Input/Output complexity of sorting and related problems. *Communications of the ACM*, **31**:1116–1127, 1988.

[3] A. Averbuch, R. Coifman, D. Donoho, M. Elad, and M. Israeli. Fast and accurate polar Fourier transform. *Appl. Comput. Harmon. Anal.*, 21:145–167, 2006.

[4] G. Beylkin. On the fast Fourier transform of functions with singularities. *Appl. Comput. Harmon. Anal.*, **2**(4):363–381, 1995.

[5] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, **19**(90):297–301, 1965.

[6] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. on Sci. Comput.*, **14**(6):1368–1393, 1993.

[7] J. A. Fessler and B. P. Sutton. Non uniform fast Fourier transforms using min-max interpolation. *IEEE Trans. Sig. Proc.*, **51**(2):560–574, 2003.

[8] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, **93**(2):216–231, 2005.

[9] L. Greengard and J. Y. Lee. Accelerating the nonuniform fast Fourier transform. *SIAM Review*, **46**(3):443–454, 2004.

[10] J. I. Jackson, C. H. Meyer, and D. G. Nishimura. Selection of a convolution function for Fourier inversion using gridding. *IEEE Trans. on Med. Imaging*, **10**(3):473–478, 1991.

[11] Q. H. Liu and N. Nguyen. An accurate algorithm for nonuniform fast Fourier transforms (NUFFTs). *IEEE Microwave and Guided Wave Letters*, **8**(1):18–20, 1998.

[12] N. Nguyen and Q. H. Liu. The regular Fourier matrices and nonuniform fast Fourier transforms. *SIAM Journal on Scientific Computing*, **21**(1):283–293, 1999.

[13] J. Song and Q. H. Liu. An efficient MR image reconstruction method for arbitrary k-space trajectories without density compensation. *Engineering in Medicine and Biology Society*, pages 3767–3770, 2006.