README

The package contains functions for computing the

1. 3D forward and inverse pseudo-polar Fourier transform,
2. 3D forward and inverse discrete Radon transform.

The package contains implementations of three inversion algorithms for the pseudo-polar Fourier transform:

1. A reference iterative implementation, described in the paper “A. Averbuch and Y. Shkolnisky. 3D Fourier based discrete Radon transform. Applied and Computational Harmonic Analysis, 15(1):33-69, 2003.”
2. A fast inversion, described in “Fast convolution based inversion of the pseudo-polar Fourier transform”, Y. Shkolnisky and S. Golubev, submitted.
3. A GPU implementation of the code in 2.
4. An implementation of a direct inversion, described in “A. Averbuch, G. Shabat and Y. Shkolnisky. Direct inversion of the 3D pseudo polar Fourier transform. SIAM Journal on Scientific Computing, 38 (2), A1100-A1120, 2016” (Highlights: running time independent of the input, no convergence criterion is required by the algorithm).

INSTALLATION

1. Compile MEX files:
    a. From MATLAB, CD to the directory into which you have extracted the PPFT3 package.
    b. Run the script “makemex.m”
    c. Note that this script should be executed only once to compile MEX files.
2. Every time you want to use the package, run first the script “initpath.m”.

DIRECTORY STRUCTURE

1. forward – contains the forward transforms of the 3D ppft and Radon transforms
2. refcode – Reference code for the 3D pseudo-polar Fourier transform. The functions in this directory are implemented exactly as described in the paper “A. Averbuch and Y. Shkolnisky. 3D Fourier based discrete Radon transform. Applied and Computational Harmonic Analysis, 15(1):33-69, 2003”, without any optimizations.

3. Legacy – An implementation of the inverse pseudo-polar Fourier transform based on conjugate-gradients iterations, where each iteration uses the forward and adjoint transforms. While this algorithm is asymptotically as fast as the others below, it is much slower in practice.
4. conv_inverse – Fast convolution based inversion of the pseudo-polar Fourier transform.
5. conv_inverse_gpu - GPU implementation of the code in "conv_inverse".
6. direct_inverse – Fast direct inversion of the pseudo-polar Fourier transform.
7. nufft – Non-uniform Fast Fourier transform, used by the direct inversion code. Can be downloaded from http://www.cims.nyu.edu/cmcl/nufft/nufft.html
8. common – Files that are common to the various routines
9. tests – Various functions to test the accuracy and speed of the forward and inverse transforms.

IMPORTANT FUNCTIONS

Note that all functions are documented using MATLAB style documentation.

ppft3	Forward 3D pseudo-polar Fourier transform.
ippft3	Inverse 3D pseudo-polar Fourier transform.
PtP3	The gram operator of the preconditioned pseudo-polar Fourier transform, namely a consecutive application of the forward transform followed by its preconditioned adjoint.
./conv_inverse/ippft3_conv	Optimized inverse of the 3D pseudo-polar Fourier transform. Based on expressing the Gram operator of the transformation as a convolution.
./conv_inverse/FPtP	Optimized implementation of the (preconditioned) Gram operator of the transformation. Based on writing the Gram operator as a convolution.
./conv_inverse_gpu/ippft3_gpu	GPU implementation of ./conv_inverse /ippft3_conv
./direct_inverse/ippft3_direct	Optimized implementation of the direct inverse of the transform (as opposed to the other inversion algorithms which are iterative)

CONVOLUTION-BASED INVERSION

The optimized inversion of the pseudo-polar Fourier transform is based on expressing the Gram operator of the PPFT as a convolution. Fast application of this Gram operator requires pre-computing a filter that corresponds to the convolution.

The first time the inversion function "./conv_inverse/ippft3_conv" is called for a given size n, the required filter is computed and stored. This stored filter will be loaded and used on subsequent calls of ippft3_conv for the same n.

The pre-computation of the filter is an O(n log n) procedure, but with a large constant, and will thus be slow for large n (especially for double precision accuracy). However, it needs to be computed only once. If single precision accuracy is sufficient, the required filter can be computed much faster.

For manual pre-computation of the filter, call `precomppppftfilter(n,` where nxnxn is the size of the transformed volume.

DIRECT INVERSION

The optimized direct inversion is based on an "onion-peeling procedure". The 3D pseudo-polar grid, can be thought of as a cube. In each step, a 2D surface from the side of the cube is being processed using only 1D operations. The operations are done efficiently by utilizing the properties of Toeplitz matrices and non-uniform Fourier transform operators.

TIMINGS

PERFORMANCE OF CONVOLUTION-BASED INVERSION

The following table was generated by running tests/testippft on a dual CPU Xeon 5560 2.8GHz (total of 8 cores).

The columns in the following table are:

err_ref	Reconstruction error for the reference inverse PPFT implementation ippft3_ref.
err_ppft3	Reconstruction error for the optimized inverse PPFT implementation ippft3.
err_conv	Reconstruction error for the convolution-based inverse PPFT implementation fippft3.
t_ref	Timing (in seconds) for the reference inverse PPFT implementation ippft3_ref.
t_ppft3	Timing (in seconds) for the optimized inverse PPFT implementation ippft3.
t_conv	Timing (in seconds) for the convolution-based inverse PPFT implementation fippft3.

All timings are given in seconds.

n	err_ref	err_ppft3	err_conv	t_ref	t_ppft3	t_conv	t_ref/t_ppft3	t_ref/t_conv
4	1.73E-08	1.73E-08	1.73E-08	2.272	0.541	0.028	4.2	82.29
8	8.32E-10	8.32E-10	8.32E-10	9.687	1.73	0.084	5.6	115.89
16	8.43E-11	8.43E-11	8.13E-11	47.371	6.431	0.306	7.37	154.64
20	1.18E-11	4.12E-11	4.31E-11	94.43	9.96	0.572	9.48	165.13
32	3.86E-12	3.86E-12	3.89E-12	253.435	31.736	2.116	7.99	119.79
40	1.23E-12	1.23E-12	1.34E-12	434.538	61.058	3.965	7.12	109.61
64	1.97E-13	1.97E-13	3.93E-13	1443.9	155.783	18.946	9.27	76.21

PERFORMANCE OF GPU-BASED INVERSION

The following tables were generated by running tests/testippft_gpu on a dual CPU Xeon 5560 2.8GHz (total of 8 cores), using NVIDIA GTX TITAN, and MATLAB R2013b.

FILTER PRECOMPUTATION

Time required to precompute the inversion filter for various values of n. t_single is for single precision filter, t_double is for double precision filter. All timings are given in seconds.

n	t_single	t_double
16	0.968	10.528
32	5.104	82.915
50	11.352	174.106
64	33.938	600.129
80	53.893	1094.426
100	88.410	1589.688
128	342.817	5409.995

INVERSION ACCURACY

Relative error of the inverse pseudo-polar Fourier transform. Error is measured by taking the forward PPFT followed by the inverse, and computing the relative error between the original and reconstructed volumes. On the GPU we use both single and double precision.

n	err_conv	err_gpu_double	err_gpu_single
16	1.225378e-10	1.224309e-10	2.444791e-06
32	4.158938e-12	5.442280e-12	2.407136e-06
50	6.326765e-13	6.327108e-13	2.232903e-06
64	3.881366e-13	3.880230e-13	2.236985e-06
80	6.070970e-13	6.071182e-13	2.122879e-06
100	1.780523e-12	1.780522e-12	2.020462e-06
128	1.384762e-12	1.384767e-12	2.121250e-06

INVERSION TIMINGS

The columns in the following table are:

t_conv	Timing (in seconds) for the convolution-based inverse PPFT implementation fippft3.
t_gpu_single	Timing (in seconds) for the GPU implementation of the convolution-based inverse PPFT fippft3_gpu in single precision.
t_gpu_double	Timing (in seconds) for the GPU implementation of the convolution-based inverse PPFT fippft3_gpu in double precision.

All timings are given in seconds.

n	t_conv	t_gpu_double	t_gpu_single	t_conv/t_gpu_double	t_conv/t_gpu_single
16	0.263	0.201	0.197	1.31	1.34
32	1.638	0.689	0.639	2.38	2.56
50	7.185	2.209	1.906	3.25	3.77
64	16.637	3.754	3.329	4.43	5.00
80	29.448	6.837	5.816	4.31	5.06

100	60.600	13.490	11.298	4.49	5.36
128	152.136	23.984	21.318	6.34	7.14

PERFORMANCE OF DIRECT INVERSION

Size	Time (sec)	Error (relative MSE)
64x64x64	7.43	1.69e-15
128x128x128	40.15	3.6e-15
256x256x256	270	1.25e-14

February 7, 2016

Yoel Shkolnisky
Gil Shabbat