Parallel-Computing Reconstruction for Parallel Encoding with Parallel Imaging

Introduction Several systems for image encoding with nonlinear, non-bijective magnetic fields have been physically realised [2] [3]. In complement, formulation of a reconstruction procedure to interpret data acquired by such systems has been presented [1] ('structured PatLoc reconstruction'). In this, the SENSE technique [4] distinguishes the encoding ambiguity of non-bijective fields, and spatial resampling yields intelligible images for display. Both SENSE and resampling are suited to GPU-computing architecture. Here it is shown that a modified reconstruction procedure implemented in a parallel programming paradigm yields impressive computational acceleration, even for data from many receive coils and for large images.

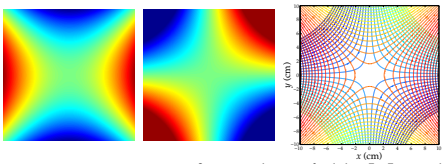
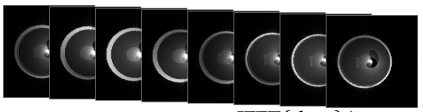
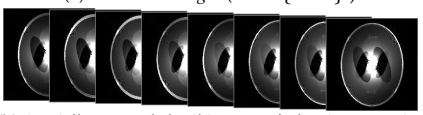


Figure 1: *Maps of encoding fields* [1] *and 'map' of resolution of the fields: pixel sizes are areas between lines, and alike-colour-enclosed regions map to the same encoded position.*

Theory The reconstruction [1] is a multi-stage procedure, outlined as (1) IDFT, (2) SENSE reconstruction in encoding spaces, (3) spatial-sample-density intensity correction in encoding spaces and (4) spatial resampling. The encoding spaces comprise partitions of the spatial domain into 'sub-domains' over which the encoding fields are bijective (e.g., top and bottom halves in fig. 1), so that the SENSE operation sorts pixel values into each warped partition. Computing SENSE reconstruction in encoding spaces requires 'warping' the sensitivity maps, and intensity correction uses the Jacobian of the encoding fields. Stages (2)-(4) are a set of operations of the form, 'for each PIXEL in X, do TASK to PIXEL'. This problem is 'embarrassingly parallel'! The massively parallel nature of the operations is appropriate for implementation by GPU computing. Additionally, resampling (interpolation) is a task well-suited to the textured memory GPU architecture—a speed advantage beyond parallelisation of computation.



(a) *'Raw' coil images* (IFFT{data})



(b) *Spatially-resampled coil images w/o density correction*

Figure 2: *Raw images evidence clearly the warping due to field nonlinearity, but the non-bijective field-induced aliasing is not evident until spatially resampled to a uniform grid.*

Methods The approach outlined above is modified by transposing the reconstruction and spatial resampling stages. Then the resampling does not require partition of the spatial domain, and the sensitivity maps do not need to be warped. (Intra-pixel error results from resolution mismatch between image and sensitivity map but is not significant.) The field-induced encoding is many-to-one, the resampling is one-to-many, and the resampled images demonstrate obvious aliasing. The quadratic fields give a position and its 180° rotation about the origin the same frequency, so this is the aliasing evident in the uniform-grid resampling (fig. 2b). The final reconstruction (fig. 3) is assembled from approximate solution of the coil sensitivity equation for all such pairs of pixels, solving the normal equations $(C^*C)\,\hat{m} = C^* m_C$ (as opposed to forming the pseudo-inverse $C^\dagger = (C^*C)^{-1}\,C^*$) for efficiency and computational stability.

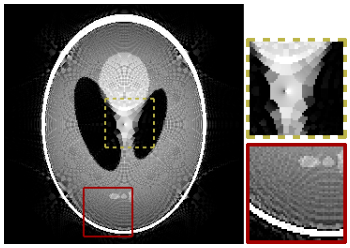


Figure 3: *Reconstructed image; enc. fields are not linear, so resolution is not uniform!*

To demonstrate advantages of parallel implementation, reconstructions were timed for single-CPU programs written in Matlab and in C and for a GPU program written in C with CUDA. Data for reconstruction are generated by specifying a spin-warp pulse sequence to modulate the encoding fields; each voxel of a numerical phantom takes phase according to its position r and the time t as $\int_0^t \gamma B(r,\tau)d\tau$, and the aggregate signals for each coil are sampled during the conventional spin-warp readout time. Solution of normal equations is implemented in Matlab by the \ ('backslash') operator and in C by LU decomposition with forward- and back-substitution. Correctness of the computations is verified by RMS difference between reconstructions.

Reconstruction by Matlab, C, and CUDA programs are (respectively) timed using `tic` and `toc` functions, `MPI_Wtime`, and CUDA event timers. Computations are performed by AMD Opteron 2384 (2.7GHz Quad-Core, 512K/6M L2/L3 cache) with 32GB RAM, and by Nvidia Tesla M2050 (1.15GHz, 3 GB memory, CC 2.0). The CPU programs (Matlab, C) use 1 core. Timings for GPU computation include all device 'overhead' (memory alloc., data transfer, etc.), and all program times exclude disk access.

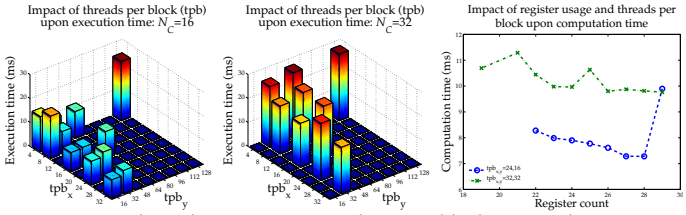


Figure 4: *GPU kernel optimisation—choosing block size and register usage to increase occupancy—increases speed-up. The problem size affects speed-up attained, but the trend is the same, and block size can be chosen accordingly.*

	Reconstruction time (s)			Speed-up	
(N_{px}, N_c)	CPU (M)	CPU (C)	GPU	CPU (C)	GPU
(262144, 16)	4.615230	0.674928	0.015125	6.8381	305.14
(262144, 32)	4.826435	1.272207	0.031572	3.7937	152.87
(400896, 16)	7.006817	0.939871	0.033140	7.4551	211.43
(400896, 32)	7.466263	1.699430	0.048214	4.3934	154.86

Results The maximum RMS difference between any pair of reconstructions is 4.8×10^{-7}. Time to reconstruct 512^2- and 576×696-pixel images from 16- and 32-coil data are impressively fast. Some GPU kernel optimisations have been used (e.g., block size, register usage) and may account for up to a factor-of-10 improvement (fig. 4). The speed difference between C and Matlab is not surprising but is not significant: the Matlab reconstruction is not compiled, and GPU computation with Matlab is also feasible.

References [1] Schultz et. al, MRM 64:447-56, 2010. [2] Gallichan et. al, MRM 65:702-14, 2011. [3] Stockmann et. al, ISMRM '12:717. [4] Pruessmann et. al, MRM 42:952-962, 1999.

Discussion The degree of acceleration is largely determined by parameters of the problem size. For a relatively small number of coils, a majority of the speed-up results from GPU parallel implementation of the resampling. However, for higher coil-counts, parallelisation of the matrix multiplications account more for the speed gains. It is noted, but not demonstrated, that a higher degree of aliasing—which results from encoding by higher-order (in the spherical harmonic-order sense) fields—increases the benefit of parallel LU and substitution.