Parallelized reconstruction of spiral Fourier velocity encoding MRI data in multicore processors

Rosana Ribeiro Lima[1], and Joao L. A. Carvalho[1]
[1]*Department of Electrical Engineering, University of Brasília, Brasília, DF, Brazil*

Introduction: Fourier velocity encoding (FVE) may be useful in the assessment of valvular disease [1] and of carotid wall shear stress [2], as it eliminates partial volume effects that are an issue in phase-contrast imaging. Although the scan-time of 2DFT FVE is prohibitively long for clinical use, the spiral FVE method [1] is substantially faster. The scan time in FVE can be significantly reduced using compressed sensing [3]. However, iterative reconstruction of spiral FVE is time-consuming, due to its multidimensionality and the use of non-Cartesian sampling. The sequential implementation of spiral FVE reconstruction algorithms underuses the total capacity of multicore processors. In this work, we demonstrate a reduction of reconstruction time of spiral FVE data through parallel programming.

Spiral FVE: Datasets consist of temporally-resolved stacks-of-spirals in k_x-k_y-k_v space [1]. A non-uniform FFT (NUFFT) [4] along k_x-k_y, followed by an FFT along k_v, produces the spatio-temporal-velocity distribution, $m(x,y,v,t)$. If multi-slice or 3D acquisitions are used, the dataset is five-dimensional: $m(x,y,z,v,t)$.

Parallelized reconstruction: Matlab's **parfor** function implements a parallelized "for" loop, which can execute in parallel on a multicore processor. Five types of variables can be used inside a "parfor" loop: temporary (created inside the loop; not used outside it); broadcast (defined before the loop; used inside the loop, but never assigned); loop (loop index for arrays); reduction (accumulates a value across iterations of the loop, regardless of iteration order); and sliced (an array whose segments are operated on by different iterations of the loop; may be indexed only by the loop variable or by broadcast variables) [5]. If a sliced array is operated within a nested "for" loop, it cannot be used again inside the "parfor" loop (see Fig. 1).

Methods: Multi-slice CINE spiral FVE scans were performed on a GE Signa 3T EXCITE HD system (40 mT/m, 150 T/m/s gradients), using a 4-channel carotid coil. Scan parameters: 1.4×1.4×5 mm^3 spatial resolution (8×1012-sample variable-density spiral readouts), 5 cm/s velocity resolution (32 velocity encodes), 12 ms temporal resolution (43 cardiac phases), 5 axial slices, 146-second acquisition per slice (256 heartbeats at 105 bpm). Results from one slice are shown in Fig. 2. The data were reconstructed in Matlab, using the non-uniform FFT toolbox by Fessler JA. The reconstruction algorithms were rewritten according to the above guidelines. Reconstruction times were evaluated on dual- and quad-core processors, on Matlab 2008a and 2011a, and for datasets of different dimensionalities: conventional 2D spiral imaging, single-slice spiral FVE (4D), and multi-slice spiral FVE (5D). For multi-slice reconstruction, the order of the z and t loops was exchanged, so that sliced variables were indexed only by the loop variable. The initialization of temporary matrices was moved into the loop; this makes these matrices temporary variables instead of broadcast variables, allowing their values to change inside the loop. For reconstruction of 2D spiral data, portions of the k-space data were separately reconstructed on each iteration of a "parfor" loop.

Results and Conclusion: The use of parallel computing considerably increased CPU usage (Fig. 3) and reduced reconstruction time (Table 1). Greater reduction was observed with the quad-core CPU. Parallel computing was able to reduce reconstruction time of single-slice spiral FVE by 43% in Matlab 2008a and by 38% in Matlab 2011a, in a quad-core processor. Matlab 2011a presented significantly faster reconstruction times, and the reduction achieved from using parallelized reconstruction was less significant, because newer versions provide improved multicore support for many functions. This increases CPU usage even within traditional "for" loops. Parallel computing was unable to reduce reconstruction time for the small datasets, because of initialization overhead.

```
a                                    b
X = zeros(3,12);                     X = zeros(3,12);
parfor m = 1:3                       parfor m = 1:3
    for n = 1:12                         p = zeros(1,12);
        X(m,n) = m + n;                  for n = 1:12
    end                                      p(n) = m + n;
    disp(X(m,1))                         end
end                                      disp(p(1))
                                         X(m,:) = p;
                                     end
```

Fig. 1: Example of (a) incorrect and (b) correct implementations of a "parfor" loop. In (a), the sliced variable X was defined inside a nested "for" loop, therefore it cannot be referenced again inside the "parfor" loop [5]. In (b), this is addressed by using a temporary array p.

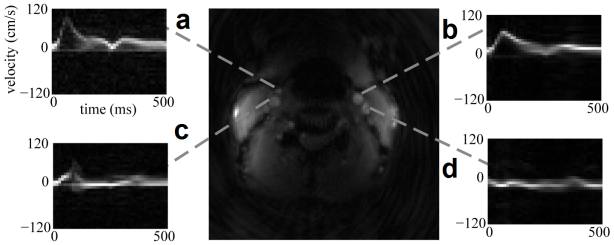

Fig. 2: Time-velocity distributions for select voxels from an axial slice prescribed at the neck of a healthy volunteer: (a) right external carotid artery, (b) left carotid bifurcation; (c) right internal carotid artery; and (d) left jugular vein.

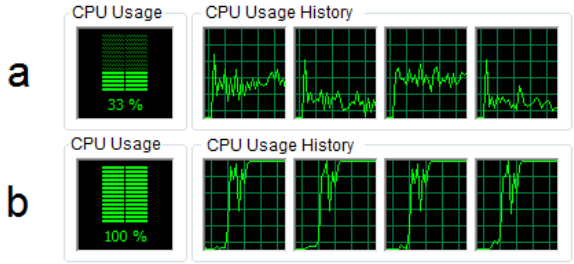

Fig. 3: CPU usage for a quad-core processor while running (a) a sequential implementation of the reconstruction algorithm (for); and (b) a parallelized implementation (parfor).

Acknowledgment: R. R. Lima received a ProIC fellowship from CNPq. Imaging was performed at USC in collaboration with Prof. Krishna Nayak.

References: [1] Carvalho JLA and Nayak KS. MRM 57:639, 2007. [2] Carvalho JLA et al. MRM 63:1537, 2010. [3] Gamper U et al. MRM 59:365, 2008. [4] Fessler JA and Sutton BP. IEEE TSP 51:560, 2003. [5] http://www.mathworks.com/help/toolbox/distcomp/brdqtjj-1.html

Table 1: Comparison of reconstruction times (in seconds) for the sequential algorithm ("for" loop) and the parallelized approach ("parfor" loop).

	MATLAB R2008a						MATLAB R2011a					
	dual-core processor			quad-core processor			dual-core processor			quad-core processor		
	for	parfor	reduction	for	parfor	reduction	for	parfor	reduction	for	parfor	reduction
$m(x,y)$	0.8	1.1	no	0.3	0.9	no	0.7	1.0	no	0.3	0.9	no
$m(x,y,v,t)$	171.2	107.8	37%	87.5	49.8	43%	107.1	90.3	16%	65.2	40.1	38%
$m(x,y,z,v,t)$	***	***	***	435.7	275.5	37%	***	***	***	334.9	218.3	35%

*** *The five-dimensional dataset was not reconstructed with the dual-core computer due to insufficient RAM memory.*