RECONSTRUÇÃO PARALELIZADA DE DADOS DE RESSONÂNCIA MAGNÉTICA DE FLUXO EM PROCESSADORES MULTI-NÚCLEO

Rosana R. Lima e João L. A. Carvalho

Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, Brasil

e-mail: joaoluiz@pgea.unb.br

Abstract: Multidimensional magnetic resonance imaging (MRI) protocols based on non-Cartesian sampling are very useful in the analysis of cardiovascular blood flow dynamics and/or cardiac function, by providing reduced scan time. However, reconstruction of multidimensional and non-Cartesian MRI data involves high computational cost. This paper presents a simple and practical approach to reduce the reconstruction time of these data, through parallelized multi-core processing. For a demonstration, we use spiral Fourier velocity encoding (FVE) data. Spiral FVE is a multi-dimensional a non-Cartesian method for rapid flow MRI assessment, which provides velocity distribution measurements which are fully resolved both in space (in three dimensions) and time.

***Palavras-chave*: reconstrução paralelizada, multi-núcleo, processamento paralelo, ressonância magnética, FVE**

Introdução

A visualização do fluxo sanguíneo nas carótidas é importante na avaliação de pacientes com estenose carotídea. A técnica mais utilizada para a quantificação do fluxo carotídeo é a ultrassonografia. Trata-se de um exame barato, que utiliza um equipamento portátil e permite boa resolução espaço-temporal em tempo real. Entretanto, para pacientes com muito tecido adiposo, cicatrizes de cirurgias ou tecido ósseo no percurso do feixe de ultrassom, a qualidade do exame pode ser comprometida. Ainda, em caso de não alinhamento da sonda com o vaso a ser analisado, é necessário estimar o ângulo de insonação para corrigir apropriadamente as velocidades medidas. Tal estimação depende da habilidade do operador, o que pode resultar em medidas imprecisas do fluxo [1]. Uma alternativa ao ultrassom é a ressonância magnética (RM), que por sua vez tem potencial para fornecer um exame cardiovascular completo, incluindo a medição do fluxo sanguíneo.

Essencialmente, existem duas formas de se medir fluxo por RM: utilizando contraste de fase, que usa duas aquisições com gradientes bipolares com polaridades opostas, ou utilizando a codificação de velocidade em Fourier (*Fourier velocity encoding*, ou FVE) [2], que usa múltiplas aquisições com gradientes bipolares com amplitudes diferentes para codificar a distribuição de velocidades, $s(v)$, no domínio de Fourier, $S(k_v)$. A primeira é a mais utilizada, devido ao tempo de aquisição reduzido; no entanto, ela não oferece precisão quando, em um pixel da imagem, há *spin*s com velocidades diferentes (volume parcial) [3]. Isso é muito comum em jatos de fluxo devido a uma estenose, por exemplo. A segunda resolve esse problema, ao medir a distribuição de velocidades em cada pixel. Entretanto, a aquisição pelo método da FVE pode ser demorada, o que pode ser amenizado implementando a codificação espacial por meio de aquisições em espiral [4].

O uso de aquisições com amostragem não-Cartesiana, como é o caso das aquisições em espiral, gera outro problema, desta vez com relação à complexidade computacional da reconstrução. No caso de dados multidimensionais, como os de FVE, o uso de amostragem não-Cartesiana resulta em longos tempos de reconstrução.

Contudo, sabe-se que a tradicional reconstrução sequencial subutiliza a CPU quando se têm múltiplos núcleos ou mais de um processador. Para reduzir o tempo de reconstrução dos dados, este trabalho propõe o uso de múltiplos núcleos de processadores em sua capacidade máxima na reconstrução, por meio da reconstrução paralelizada, o que pode ser feito com pequenas alterações no algoritmo utilizado.

Teoria

Espaço-k – Imagens de RM tipicamente correspondem à distribuição espacial de núcleos ^{1}H em um plano: $s(x,y)$. Enquanto a técnica de RM por contraste de fase mede um mapa de velocidades, $v(x,y)$, a técnica de codificação de velocidades por Fourier mede a distribuição de velocidades $s(v)$ associada a cada pixel. O conjunto de dados adquiridos é uma matriz $S(k_x,k_y,k_v)$, que é o espaço-k (transformada de Fourier) associado à distribuição $s(x,y,v)$ [2,4].

Aquisição – A técnica de FVE com codificação espacial em espiral utiliza múltiplos gradientes bipolares escalonados (ao longo do eixo z) para a codificação da distribuição de velocidades no domínio de Fourier (k_v) e gradientes oscilatórios que codificam a informação espacial através de trajetórias espirais ao longo do espaço k_x-k_y (Figura 1). Assim, o espaço-k associado é uma pilha de espirais em k_x-k_y-k_v (Figura 2).

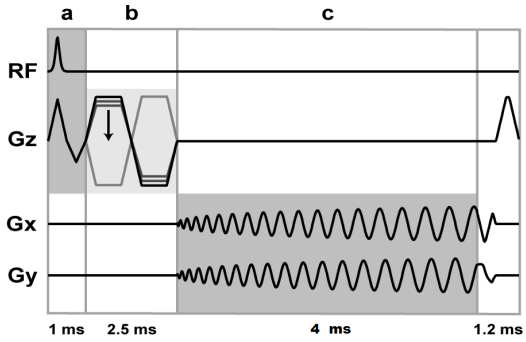

Figura 1: Sequência de pulsos da técnica de codificação de velocidades por Fourier com aquisições em espiral: (a) seleção de corte, (b) gradiente de codificação de velocidades, (c) gradientes de codificação espacial.

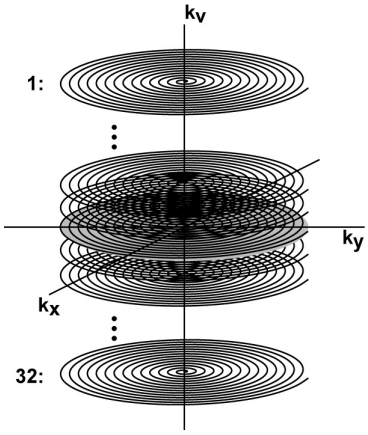

Figura 2: Trajetória da técnica no espaço-k. Os dados correspondem a pilhas de espirais em k_x,k_y,k_v, que são a transformada de Fourier de distribuições $s(x,y,v)$. Os gradientes oscilatórios adquirem espirais em k_x,k_y, enquanto os gradientes bipolares codificam em k_v.

Os dados são ainda adquiridos para múltiplos cortes perpendiculares ao plano x,y (eixo z) e também para múltiplos quadros temporais (eixo t) ao longo do ciclo cardíaco, resultando na distribuição $s(x,y,z,v,t)$.

O sinal de RM pode ser medido utilizando múltiplas bobinas de aquisição. A i-ésima bobina mede o sinal $s_i(x,y,z,v,t)$. Os dados de cada bobina são reconstruídos separadamente e posteriormente combinados usando um algoritmo de soma quadrática.

Assim, os dados de FVE podem ser considerados como de seis dimensões, a saber: as três dimensões espaciais (x,y,z), velocidade ao longo do eixo z (v), instante de tempo (t) e canal ou bobina (i). Note que, após a reconstrução, a dimensão i não aparece, pois o sinal reconstruído é uma combinação dos sinais individuais de cada canal.

Reconstrução – Como os dados são adquiridos no espaço-k, é necessário aplicar uma transformada de Fourier inversa para obter as imagens. Para dados amostrados em grade Cartesiana, pode-se utilizar a transformada rápida de Fourier (FFT), que é um algoritmo extremamente rápido. Entretanto, devido ao uso de trajetórias em espiral, é necessário utilizar a transformada de Fourier não uniforme (*non-uniform Fourier transform*, ou NUFFT) [5] ao longo de k_x e k_y.

Métodos

Aquisição dos dados – Aquisições de FVE com codificação espacial em espiral foram feitas em um equipamento GE Signa 3T EXCITE HD (gradientes com amplitude máxima de 40 mT/m e taxa máxima de variação de 150 T/m/s), utilizando uma bobina para carótidas de 4 canais. Os parâmetros da aquisição foram: 1,4×1,4×5 mm^3 de resolução espacial (8 leituras de 1012 amostras cada, usando espirais de 4 ms com densidade variável), 5 cm/s de resolução de velocidade (32 amostras ao longo de k_v), 12 ms de resolução temporal (43 quadros temporais), 5 cortes axiais, 146 segundos totais de aquisição por corte (256 batimentos cardíacos a 105 bpm). O comitê de ética da University of Southern California aprovou o protocolo utilizado, e os voluntários assinaram consentimento informado.

Paralelização da reconstrução – Para paralelizar o algoritmo de reconstrução, utilizou-se a função *parfor* do Matlab (Mathworks, Inc., South Natick, MA, EUA). Para fins de comparação, foram utilizadas tanto a versão R2008a quanto a R2011a do pacote de software. Um laço *parfor* (*parallel for*) tem a mesma função de um laço *for* convencional, mas é capaz de otimizar a execução por meio da distribuição das iterações do loop entre os vários núcleos da CPU.

Para que a paralelização do algoritmo de reconstrução fosse possível, algumas alterações foram necessárias, devido a algumas restrições impostas pela função *parfor*. Somente cinco tipos de variáveis são permitidos dentro de um laço *parfor*:

a) temporária: criada dentro do laço, não podendo ser utilizada fora do mesmo;
b) *broadcast*: é definida antes do laço e pode ser usada dentro do laço, mas nunca redefinida dentro dele;
c) laço: é incrementada a cada iteração do laço, e usada para indexar vetores;
d) redução: acumula um valor a cada iteração do laço e independe da ordem de execução das iterações; e
e) *sliced*: um vetor cujos segmentos são lidos ou escritos a cada iteração do laço e que só pode ser indexado pela variável de laço ou por variáveis de *broadcast* e que, se operado dentro de um laço *for* interno, não pode ser usado novamente dentro do laço *parfor* (Figura 3).

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

Figura 3: Exemplo de uso (a) incorreto e (b) correto de um laço *parfor*. Em (a), a variável *sliced* **X** foi definida dentro do laço *for* interno, portanto não pode ser referenciada novamente dentro do laço *parfor* [5]. Em (b), isso é resolvido usando um vetor temporário **p**.

Equipamento utilizado – Os tempos de reconstrução dos algoritmos paralelizados (com laço *parfor*) foi comparada aos dos algoritmos sequenciais (com laço *for*). Os algoritmos foram avaliados em dois computadores, com as seguintes especificações:

a) *notebook* Itautec W7655 SS, com processador Intel Core 2 Duo T6500 de 2,1 GHz, 4 GB de memória RAM DDR2-800 e placa de vídeo Intel GMA4500MHD;
b) *desktop* com placa mãe Asus P5K Premium, processador Intel Core 2 Quad Q9300 de 2,50 GHz, 8 GB de memória RAM DDR2-800 e placa de vídeo Nvidia GeForce 8500 GT.

Dados bidimensionais – Primeiramente, foram reconstruídos apenas os dados correspondentes a uma imagem. Optou-se por escolher os planos onde $k_v = 0$, ou seja, em que não há gradiente para codificação do fluxo. Escolhendo um valor de instante de tempo e uma determinada fatia, realizou-se a reconstrução de $s_i(x,y)$ para cada bobina, aplicando uma NUFFT nos dados $S_i(k_x,k_y)$ e, em seguida, os dados das diversas bobinas foram combinados, resultando na imagem, $s(x,y)$.

Não foram necessárias muitas alterações no código da imagem a fim de paralelizar sua execução, pois o algoritmo era simples e as variáveis respeitavam as exigências requeridas.

Dados multidimensionais – Nos dados estudados neste trabalho, além do eixo da velocidade, tem-se a dimensão do tempo e da fatia desejada. Além disso, os dados foram adquiridos por quatro bobinas simultaneamente. Assim, temos os dados $S_i(k_x,k_y,z,k_v,t)$.

Caso só se deseje reconstruir os dados correspondentes a somente um corte axial — escolhido pelo usuário no início do programa —, teremos somente as dimensões k_x, k_y, k_v, t e i. Neste caso, para cada instante de tempo, bobina e velocidade, uma imagem $S_i(x,y,k_v,t)$ é obtida por meio da NUFFT ao longo de k_x e k_y. Em seguida, os dados das diversas bobinas pertencentes ao mesmo instante de tempo são combinados. Depois, faz-se uma transformada rápida de Fourier inversa (iFFT) ao longo de k_v para se obter a distribuição $s(x,y,v,t)$.

Para recuperar os dados em todas as suas dimensões (isto, é incluindo o eixo z, dos cortes), o processo descrito no parágrafo anterior é executado para cada corte, obtendo-se a distribuição $s(x,y,z,v,t)$.

No código para a reconstrução de um único corte (dados quadridimensionais, ou 4D), o algoritmo foi favorável à utilização do *parfor*, sem modificações significativas. Foi considerada importante, entretanto, a alocação de códigos nos laços mais externos sempre que possível, para possibilitar a redução do tempo de execução final.

No código escrito para a reconstrução dos dados pentadimensionais (ou 5D) envolvendo todos os cortes, alguns cuidados foram importantes para viabilizar o uso do *parfor*. Um deles foi com relação ao uso do *parfor* no loop mais externo. Como essa reconstrução envolve um número grande de laços, e a inicialização do *parfor* exige um tempo de viés, a reinicialização desse laço repetidas vezes é indesejável no que diz respeito ao objetivo da redução do tempo de execução do algoritmo. Outro cuidado importante é com relação ao fato de variáveis definidas fora do laço *parfor* não poderem ser alteradas dentro desse laço (a não ser quando acessadas por variáveis do tipo "laço") e, portanto, as variáveis alteradas passaram a ser do tipo "*sliced*". Assim, a matriz que recebia os valores do sinal para todos os valores de x e y para diversos valores de velocidade e bobinas precisou ser definida inicialmente dentro do laço *parfor*, a fim de se tornar uma variável do tipo "temporária", pois esta era redefinida continuamente, mas seus índices não eram variáveis do tipo "laço". As únicas variáveis com potencial de serem variáveis do tipo "laço", da maneira como o algoritmo foi escrito, são a que percorre os instantes de tempo de amostragem (quadros) e a que percorre o eixo z (cortes). Entretanto, a variável utilizada para conter os dados das imagens do pescoço mostradas ao final da reconstrução (ver Figura 7, discutida mais adiante) era redefinida continuamente dentro do laço, em função da variável de corte. Assim, esta deveria ser transformada em uma variável do tipo "*sliced*". Por isso, no laço *parfor* foi necessário utilizar a variável de corte como a variável do tipo "laço".

Resultados

A figura 4 ilustra o percentual de uso dos núcleos do processador *quad-core* com um algoritmo de reconstrução sequencial e com outro, paralelizado. Nota-se que, com as iterações do laço *for* divididas entre os núcleos, maximiza-se o uso do processador e, consequentemente, reduz-se o tempo de execução. Esta é a idéia básica da metodologia proposta neste trabalho.

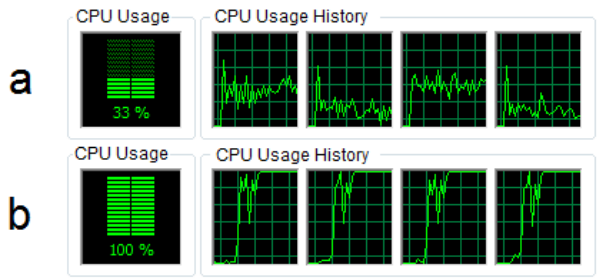

Figura 4: Percentual de uso dos núcleos do processador *quad-core* durante execução de algoritmo de reconstrução (a) sequencial e (b) paralelizado.

A reconstrução sequencial dos dados bidimensionais mostrou-se consideravelmente rápida, como mostram os resultados apresentados na Tabela 1. Isto é esperado devido ao pequeno volume de dados, quando comparado aos dados multidimensionais. Neste caso, percebe-se que a paralelização foi desfavorável, aumentando o tempo de reconstrução. Isso aconteceu porque o loop *parfor* tem um tempo viés de inicialização, que no caso não foi suficientemente compensado pela redução do tempo de reconstrução em si, devido ao pequeno volume de dados a serem processados.

Tabela 1: Tempo de execução medido (em segundos) para cada um dos tipos de reconstrução, processador e versão de Matlab

	MATLAB R2008a						**MATLAB R2011a**					
	processador *dual-core*			**processador *quad-core***			**processador *dual-core***			**processador *quad-core***		
	for	***parfor***	**redução**	***for***	***parfor***	**redução**	***for***	***parfor***	**redução**	***for***	***parfor***	**redução**
***s*(*x*,*y*)**	0.8	1.1	não	0.3	0.9	não	0.7	1.0	não	0.3	0.9	não
***s*(*x*,*y*,*v*,*t*)**	171.2	107.8	37%	87.5	49.8	43%	107.1	90.3	16%	65.2	40.1	38%
***s*(*x*,*y*,*z*,*v*,*t*)**	***	***	***	435.7	275.5	37%	***	***	***	334.9	218.3	35%

*** devido à insuficiência de memória RAM do hardware utilizado, os dados multidimensionais não foram reconstruídos no processador *dual-core*.

A Tabela 1 também mostra o resultado da reconstrução dos dados 4D. Devido ao volume maior de dados, o tempo de reconstrução, neste caso, foi significativamente reduzido com o uso do *parfor*, se comparado ao algoritmo convencional. O uso do *parfor* foi especialmente vantajoso no computador com processador *quad-core*, devido à maior possibilidade de paralelização. Após a reconstrução, a imagem bidimensional correspondente ao corte é apresentada ao usuário. Ao se clicar em um dos pixels dessa imagem, definindo assim uma posição espacial (*x*,*y*), a distribuição de velocidades resolvida no tempo, *s*(*v*,*t*), correspondente à essa posição, é apresentada (Figura 5). Pode-se então selecionar um novo pixel. As distribuições obtidas para diversos vasos sanguíneos de interesse no pescoço são apresentadas na Figura 6.

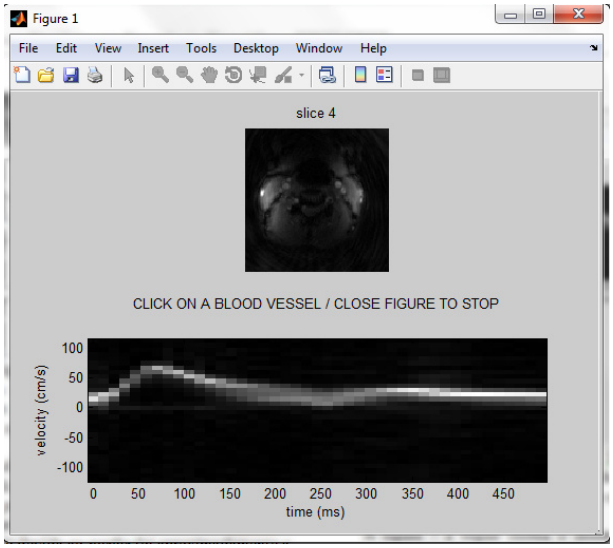

Figura 5: Resultado da reconstrução dos dados 4D de um corte. Ao se clicar em um pixel (*x*,*y*) da imagem axial do pescoço (em cima), a distribuição de velocidades resolvida no tempo, *s*(*v*,*t*), correspondente à posição espacial prescrita é apresentada (em baixo).

Os dados 5D foram reconstruídos somente no computador com processador *quad core*, devido à insuficiência da RAM do computador com processador *dual core*. No primeiro, a taxa de redução do tempo de reconstrução devido ao uso do *parfor* nas duas versões do Matlab avaliadas foi aproximadamente a mesma, a saber, de 37% para o Matlab R2008a e de 35% para o Matlab R2011a (Tabela 1). Sabendo que, o tempo de reconstrução com o algoritmo convencional é de aproximadamente 7,3 e 5,6 minutos nessas duas versões, respectivamente, consideramos a redução de tempo de reconstrução alcançada bastante significativa. Após a reconstrução, são apresentadas ao usuário as imagens *s*(*x*,*y*) correspondentes aos cinco cortes; ao se clicar em um dos cortes, é mostrada, em destaque, a imagem do corte selecionado (Figura 7). Ao se clicar em um dos pixels dessa imagem, definindo assim uma posição espacial (*x*,*y*,*z*), a distribuição de velocidades resolvida no tempo, *s*(*v*,*t*), correspondente à essa posição, é apresentada (Figura 8). Pode-se então selecionar um novo corte e/ou um novo pixel de interesse.

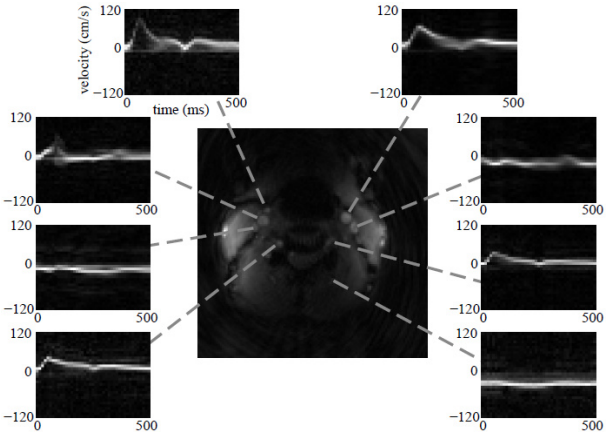

Figura 6: Distribuições de velocidades resolvidas no tempo, *s*(*v*,*t*), obtidas para diversos vasos sanguíneos de interesse no pescoço, a partir dos dados 4D de um corte.

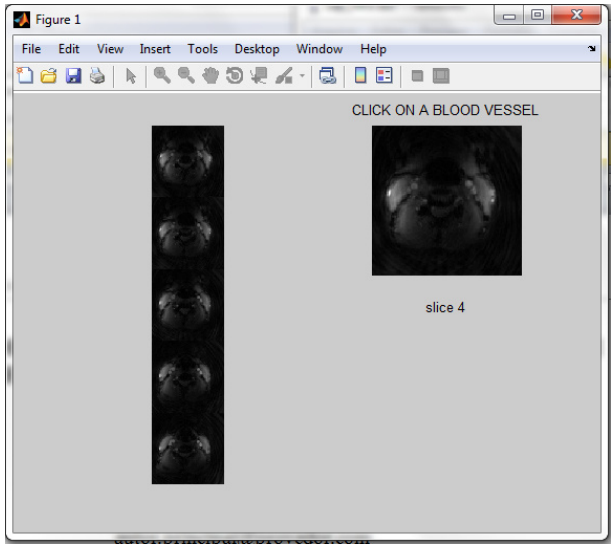

Figura 7: Após a reconstrução dos dados 5D, são apresentadas ao usuário as imagens *s*(*x*,*y*) correspondentes aos cinco cortes (esquerda). Ao se clicar em um dos cortes, é mostrada, em destaque, a imagem do corte selecionado (direita), para que seja definida a localização espacial do fluxo de interesse.

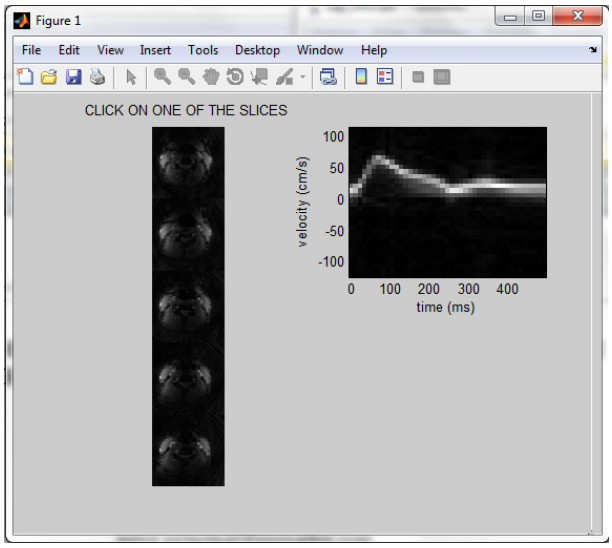

Figura 8: Para os dados 5D, após definido o corte e o pixel de interesse (ver Figura 7), a distribuição de velocidades resolvida no tempo, $s(v,t)$, correspondente à posição espacial selecionada, é apresentada.

Conclusão

Os resultados apresentados têm o objetivo de motivar pesquisadores na área de imagens médicas a, sempre que necessário e possível, recorrer a métodos de reconstrução paralelizada, visto que estes são de fácil implementação e podem proporcionar grande economia de tempo de reconstrução e processamento. Os benefícios são especialmente significativos ao se lidar com volumes grandes de dados (ex: dados multidimensionais e/ou de alta resolução) e quando se usa processadores com um grande número de núcleos.

Mostrou-se ainda que o Matlab apresenta, em suas versões mais novas, uma tendência a incorporar a paralelização — até mesmo por meio de processamento utilizando a placa gráfica — em suas funções *built-in*, diminuindo seu tempo de execução mesmo quando laços *for* tradicionais são utilizados. Ainda assim, a paralelização por *parfor* se mostrou um recurso valoroso. Viu-se ainda que, tanto para o algoritmo sequencial quanto para a implementação com *parfor*, o computador com processador *quad-core* apresentou um tempo de reconstrução menor que o computador com processador *dual-core*, sendo que, quando a paralelização do algoritmo foi empregada, o tempo total de reconstrução foi ainda menor. Isso ocorre porque, quando se utiliza a função *parfor*, as iterações do laço *for* são distribuídas entre os vários núcleos do processador, maximizando o uso dos mesmos e assim reduzindo o tempo de processamento. Estas lições podem ser úteis em diversas aplicações envolvendo processamento ded dados multidimensionais volumosos, mesmo fora da área de imagens médicas.

Agradecimentos

Este trabalho recebeu apoio financeiro oriundo do Edital MCT/CNPq Nº 014/2010 – Universal. R. R. Lima é aluna de iniciação científica do ProIC/DPP/UnB com bolsa financiada pelo CNPq. A aquisição dos dados foi realizada na University of Southern California em colaboração com o Prof. Krishna S. Nayak.

Referências

[1] Hoskins, P.R. (1996), “Accuracy of maximum velocity estimates made using Doppler ultrasound systems”, *Br J Radiol*, v. 69, n. 818, p. 172–7.

[2] Moran, P.R. (1983), “A flow velocity zeugmatographic interlace for NMR imaging in humans”, *Magn Reson Imaging*, v.1, n. 4, p.197-203.

[3] Tang, C., Blatter, D. D. and Parker, D. L. (1993), “Accuracy of phase-contrast flow measurements in the presence of partial-volume effects”, *Journal of Magnetic Resonance Imaging*, v. 3, n. 3, p. 377-85.

[4] Carvalho, J. L. A. and Nayak, K. S. (2007), “Rapid quantitation of cardiovascular flow using slice-selective Fourier velocity encoding with spiral readouts”, *Magnetic Resonance in Medicine*, v. 57, n. 4, p.227-88.

[5] Fessler, J. A. and Sutton, B. P. (2003), “Nonuniform fast Fourier transforms using min-max interpolation”, In: *IEEE Transactions on Signal Processing* v. 51, 560-74.