Anais XIV Simpósio Brasileiro de Sensoriamento Remoto, Natal, Brasil, 25-30 abril 2009, INPE, p. 6935-6942.

Segmentação multiresolução: uma abordagem paralela para segmentação de imagens de alta resolução em arquiteturas de múltiplos núcleos

Patrick Nigri Happ [1]
Rodrigo da Silva Ferreira [1]
Cristiana Bentes [1]
Gilson Alexandre Ostwald Pedro da Costa [2]
Raul Queiroz Feitosa [2]

[1] Universidade do Estado do Rio de Janeiro - UERJ
Rua São Francisco Xavier, 524 - 20550-900 - Maracanã - RJ, Brasil
{patrick.happ, rodrigosilferreira, cristianabentes}@gmail.com

[2] Pontifícia Universidade Católica do Rio de Janeiro – PUC-RJ
Caixa Postal 38097 - 22453-900 - Gávea - RJ, Brasil
{gilson, raul}@ele.puc-rio.br

Abstract. Segmentation is a crucial step in automatic image interpretation. However, many of existing segmentation algorithms have high computational cost for large images. The main focus of this paper is to tackle this problem by using parallel processing. The idea is to explore current multicore architectures available in commercial processors in order to speedup the segmentation process. A multithreading parallel implementation of a region growing algorithm is proposed that aims at providing better execution times, while delivering the same outcome produced by the sequential algorithm version. The algorithm is able to work with any number of threads, which is defined as an input parameter, so as to take full advantage of the upcoming processors having any number of cores. The current parallel implementation was tested on three different images on a quad-core processor and obtained up to 2.3 of segmentation speedup.

Palavras-chave: remote sensing, image processing, parallel processing, sensoriamento remoto, processamento de imagens, processamento paralelo.

1. Introdução

A segmentação de imagens tem sido um problema amplamente discutido no campo de processamento de imagens digitais e visão computacional. Algoritmos de segmentação por crescimento de regiões agrupam *pixels* ou sub-regiões em regiões maiores, partindo de um conjunto de pontos iniciais (sementes) que crescem anexando regiões adjacentes que possuam propriedades similares (como, por exemplo, textura ou cor). Esta classe de algoritmos tem tido ampla aplicação especialmente na área de sensoriamento remoto. A desvantagem é o alto custo computacional associado para imagens muito grandes.

O objetivo deste trabalho é propor uma implementação paralela para o algoritmo de segmentação de imagens proposto em (Baatz et al. 2000). A idéia é aproveitar a capacidade de processamento paralelo presente na grande maioria dos processadores modernos, mais especificamente os múltiplos núcleos de computação em um mesmo processador. A solução proposta não requer, portanto, hardware especial e pode ser executada em máquinas de baixo custo disponíveis comercialmente.

A implementação paralela é baseada na divisão do trabalho em linhas de execução, ou *threads*. Uma preocupação foi manter o resultado da segmentação independente da velocidade dessas linhas de execução. O algoritmo foi introduzido utilizando a biblioteca OpenMP (Chapman, 2008) para programação com memória compartilhada e executado em um processador com quatro núcleos (*quad-core*). Obtivemos uma aceleração de até 2,3 no tempo total de execução.

O restante deste artigo está organizado da seguinte forma. Na próxima seção é apresentada uma breve descrição do algoritmo de crescimento de regiões proposto por Baatz e

Schäpe. Na seção seguinte, a proposta de implementação paralela é descrita. Na seção 4, os resultados de uma análise experimental de desempenho são apresentados e a seção 5 finaliza o trabalho com as principais conclusões e direções para trabalhos futuros.

2. Segmentação por Crescimento de Regiões

Esta seção descreve sucintamente a versão seqüencial do algoritmo de crescimento de regiões proposto por Baatz e Schäpe e utilizado no sistema Definiens (anteriormente denominado eCognition) (Definiens 2008).

Trata-se de um procedimento iterativo de otimização local, que minimiza a heterogeneidade média dos segmentos gerados. A medida de heterogeneidade usada no algoritmo possui um componente espacial e um componente espectral. A heterogeneidade espectral é definida sobre os valores das respostas espectrais dos pixels contidos num segmento. Essa medida é proporcional à média ponderada do desvio padrão de cada banda.

A heterogeneidade espacial baseia-se em dois atributos de forma: compactação e suavidade. O grau de compactação é definido como a razão entre o perímetro do segmento e a raiz quadrada de sua área (número de pixels que contém). A suavidade é definida como a razão entre o perímetro do objeto e o perímetro do retângulo envolvente mínimo (*bounding box*).

Inicialmente cada segmento representa apenas um pixel da imagem e todos os pixels estão associados a algum segmento. Os segmentos crescem na medida em que são unidos com seus vizinhos, e o menor aumento na heterogeneidade é utilizado como critério para a seleção do vizinho com o qual um segmento será unido. Para simular um crescimento paralelo, cada segmento é selecionado apenas uma vez a cada iteração.

O fator de fusão (*f*) expressa o aumento de heterogeneidade resultante da união de dois segmentos. Antes de uma operação de união, o fator de fusão é calculado para cada um dos vizinhos do segmento selecionado. O vizinho para o qual o fator de fusão for mínimo é o escolhido para a união. Contudo, a união só ocorre se o fator de fusão estiver abaixo de um determinado limiar, definido como o quadrado do parâmetro de escala, que denotaremos deste ponto do texto em diante com a letra *e*. O procedimento continua unindo segmentos até que nenhum segmento possa crescer mais.

O fator de fusão contém um componente para a heterogeneidade espectral (h_{cor}) e um componente para a heterogeneidade espacial (h_{forma}) (Equação 1). A importância relativa entre os componentes espacial e espectral é definida pelo *fator de cor* (w_{cor}).

$$f = w_{\text{cor}} \cdot h_{cor} + \left(1 - w_{\text{cor}}\right) \cdot h_{forma} \qquad (1)$$

A Equação 2 mostra a formulação da heterogeneidade espectral, onde *Obj1* é o segmento selecionado, *Obj2* é o vizinho analisado e *Obj3* é o segmento resultante da união de *Obj1* com *Obj2*. Nessa equação *c* é o índice da banda espectral e w_c é um peso arbitrário definido para a banda *c*; σ_c^{Obji} é o desvio padrão dos valores dos pixels na banda *c,* considerando todos os pixels pertencentes ao segmento *Obji*; e n_{Obji} é o número de pixels em *Obji*, para *i=1,2,3*.

$$h_{cor} = \sum_c w_c \left(n_{Obj3} \cdot \sigma_c^{Obj3} \left(n_{Obj1} \cdot \sigma_c^{Obj1} - n_{Obj2} \cdot \sigma_c^{Obj2} \right) \right) \qquad (2)$$

A heterogeneidade espacial é influenciada pelo grau de compactação do segmento e pela suavidade de sua borda (Equação 3). A medida de heterogeneidade espacial tem, portanto, dois componentes: o componente relativo à compactação h_{cmpct} e o componente de suavidade h_{suave}. A importância relativa entre estes dois componentes é definida pelo *fator de compactação*, w_{cmpct}.

$$h_{forma} = w_{cmpct} \cdot h_{cmpct} + \left(1 - w_{cmpct}\right) \cdot h_{suave} \qquad (3)$$

As Equações 4 e 5 mostram as formulações dos componentes de compactação e suavidade. Nessas equações l_{Obji} é o perímetro dos objetos e b_{Obji} o perímetro do correspondente retângulo envolvente mínimo, para $i=1,2,3$.

$$h_{cmpct} = n_{Obj3} \cdot \frac{l_{Obj3}}{\sqrt{n_{Obj3}}} - \left(n_{Obj1} \cdot \frac{l_{Obj1}}{\sqrt{n_{Obj1}}} + n_{Obj2} \cdot \frac{l_{Obj2}}{\sqrt{n_{Obj2}}} \right) \qquad (4)$$

$$h_{\text{suave}} = n_{\text{Obj3}} \cdot \frac{l_{Obj3}}{b_{Obj3}} - \left(n_{\text{Obj1}} \cdot \frac{l_{Obj1}}{b_{Obj1}} + n_{\text{Obj2}} \cdot \frac{l_{Obj2}}{b_{Obj2}} \right) \qquad (5)$$

O crescimento dos segmentos é condicionado, portanto, a um critério de heterogeneidade ajustável. Este ajuste pode ser feito pela escolha do parâmetro de escala (e), dos pesos das bandas espectrais (w_c), do fator de cor (w_{cor}) e do fator de compactação (w_{cmpct}). O ajuste no parâmetro de escala (e) influencia diretamente no tamanho dos segmentos gerados. Além disso, a relevância de cada banda espectral, a importância relativa entre forma e cor, e entre compactação e suavidade, podem ser ajustados através dos parâmetros do algoritmo.

3. Implementação Paralela

A implementação paralela do algoritmo de segmentação por crescimento de regiões proposto por Baatz e Schäpe utiliza a biblioteca OpenMP para paralelização e segue a divisão da computação em diferentes linhas de execução que compartilham a mesma área de dados na memória. A idéia principal desta implementação consiste em dividir a imagem em recortes, que serão processados por diferentes *threads,* que executam basicamente o algoritmo seqüencial de Baatz e Schäpe, além de ações de sincronização. Desta forma, cada recorte será executado simultaneamente em um núcleo de um processador com múltiplos núcleos.

Esta abordagem paralela, entretanto, enfrenta dois grandes obstáculos: (i) o tratamento dos segmentos de fronteira, ou seja, dos segmentos que possuem pelo menos um vizinho que não pertença a seu recorte; (ii) a reprodutibilidade do resultado final.

Com relação ao tratamento dos segmentos de fronteira, a principal dificuldade decorre de que as *threads* executam em paralelo, podendo, assim, ocorrer o tratamento simultâneo do mesmo segmento por mais de uma *thread*. Para evitar este problema, podem-se utilizar seções críticas (zonas na qual apenas uma *thread* pode executar de cada vez) para a atualização do segmento. A utilização deste recurso, porém, pode ter grande impacto no desempenho da segmentação, se a contenção causada pela espera por seções críticas for da mesma ordem que os ganhos obtidos pela paralelização.

Com relação à reprodutibilidade dos resultados, este é um problema inerente à temporização de cada *thread.* Em outras palavras, uma *thread* pode executar sua tarefa mais rapidamente que outra, podendo gerar ordens diferentes de visitação para os segmentos, o que afeta o resultado final da segmentação. Mesmo para a segmentação seqüencial, caso as sementes sejam visitadas em ordem diferente, o resultado da segmentação é alterado. A reprodutibilidade do resultado da segmentação é uma meta importante, por permitir que cientistas de diferentes localidades possam gerar localmente segmentações de uma mesma imagem e, assim, analisar o mesmo resultado.

Para que o processo de segmentação seja realmente independente da velocidade das *threads* e para evitar a contenção excessiva por seções críticas, os segmentos localizados nas fronteiras são tratados separadamente no algoritmo proposto. Estes segmentos são incluídos numa lista de segmentos a serem tratados posteriormente. Ao final de cada passo da segmentação (após todos os segmentos terem sido visitados), os segmentos de fronteira serão

processados de forma seqüencial. Dessa forma, o crescimento das regiões de cada *thread* será independente, não havendo necessidade de seções críticas no código.

A divisão da imagem em recortes e, conseqüentemente, a divisão do trabalho em *threads*, pode ter impacto no resultado final da segmentação. Para se alcançar um melhor desempenho em uma determinada arquitetura com múltimplos núcleos, o ideal é que a quantidade de *threads* seja sempre igual à quantidade de núcleos existentes no processador. A implementação proposta, entretanto, não faz a divisão de forma automática em função da arquitetura do processador, mas deixa o usuário defini-la. Assim, pode-se garantir a mesma divisão em recortes e a reprodutibilidade dos resultados da segmentação para diferentes arquiteturas.

3.1 Distribuição Inicial de Carga

A primeira etapa do algoritmo consiste em determinar o número de recortes a serem criados. O número de recortes corresponde à quantidade de *threads*. Caso se tenha apenas uma *thread*, o processamento será seqüencial. Para duas ou mais *threads*, a imagem será dividida em áreas distintas, conforme ilustra a Figura 1, onde cada *thread* é responsável pela análise dos *pixels* compreendidos no recorte a ela alocado.

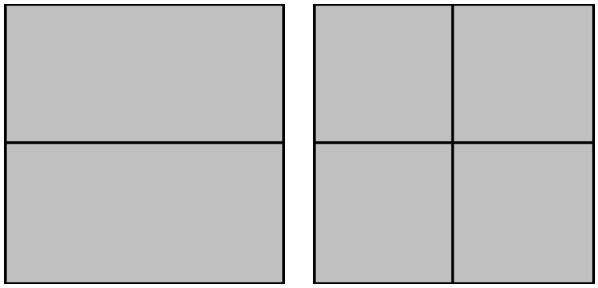

Figura 1. Divisão da imagem em retângulos para duas e quatro *threads*.

3.2 Passo de Crescimento

Após a divisão inicial da imagem, inicia-se a etapa de crescimento. Cada *thread* executa o algoritmo de crescimento de regiões de Baatz e Schäpe dentro do seu recorte. Inicialmente, a *thread* marca todos os *pixels* do recorte como segmentos a serem visitados e os organiza em uma lista de segmentos. A *thread*, então, visita cada segmento existente em sua lista e analisa o critério de heterogeneidade para cada um de seus vizinhos. Se os vizinhos forem considerados, pelo fator de fusão, como parte do segmento, eles são anexados e marcados como visitados. Caso, pelo menos um vizinho do segmento não pertença ao recorte tratado por aquela *thread*, este segmento será incluído na lista de segmentos de fronteira. Caso contrário, o segmento será processado normalmente. Este procedimento é repetido até que toda lista de segmentos de cada *thread* seja percorrida.

Ao final desta etapa, a região paralela OpenMP termina e a computação segue em uma única *thread* para o tratamento dos segmentos de fronteira. A lista de segmentos de fronteira é percorrida seqüencialmente pela *thread*, usando o mesmo processo de crescimento de regiões.

Depois que toda a lista de segmentos de fronteira é processada, a etapa inicial de crescimento termina e um novo passo se inicia. No novo passo, uma nova região paralela é iniciada. Os segmentos criados no passo anterior são considerados na lista de segmentos inicial de cada *thread* e o crescimento de regiões dentro de um retângulo é o mesmo. Este processo se repete até que nenhuma fusão ocorra num passo ou até que o número máximo de passos seja alcançado.

4. Resultados

Serão apresentados nesta seção os resultados obtidos com a implementação paralela da segmentação por crescimento de regiões. A seguir serão descritos o ambiente utilizado nos experimentos, as imagens utilizadas e em seguida o resultado da segmentação, juntamente com a avaliação do desempenho obtido com a paralelização.

4.1 Ambiente de Testes

Os experimentos foram realizados em um processador *Intel Core 2 Quad 2.40 Ghz,* com 2 Gb de memória *RAM.*

Três imagens diferentes foram utilizadas para os experimentos, sendo denominadas: *terra*, *paisagem* e *Sidney*. Os tamanhos das imagens são apresentados na Tabela 1. As imagens foram utilizadas para avaliar os ganhos de desempenho em função de seus tamanhos. As imagens s*idney* e *paisagem*, apresentadas na Figura 2, foram utilizadas também para comparação dos resultado da segmentação paralela em relação ao resultado da segmentação seqüencial.

Tabela 1. Imagens utilizadas para a avaliação de desempenho da implementação paralela.

Imagem	**Tamanho (em pixels)**
sidney	1024 x 828
paisagem	1600 x 1200
terra	3200 x 1200

Figura 2. Imagens originais antes da segmentação *sidney* (esquerda) e *paisagem* (direita).

4.2 Resultados da Segmentação Paralela

Inicialmente será analisado o resultado do processo de segmentação em função dos segmentos obtidos. Nas Figuras 3 e 4 são apresentados os resultados da segmentação das imagens s*idney* e *paisagem* para o algoritmo seqüencial de Baatz e Schape (esquerda) e para a implementação paralela (direita).

Pode-se observar nas figuras a semelhança entre os resultados da segmentação seqüencial e paralela. Não há diferenças importantes na região central, onde se encontram os segmentos pertencentes às fronteiras entre os recortes da imagem. As discrepâncias no resultado da segmentação se devem exclusivamente à diferença na ordem em que os segmentos são processados.

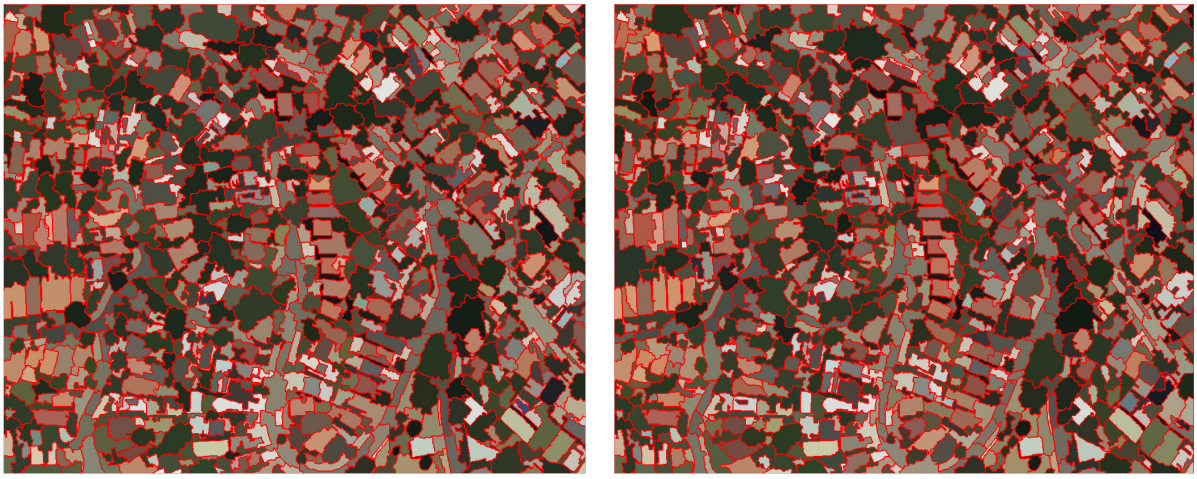

Figura 3. Resultado da segmentação de sidney pelo algoritmo seqüencial (esquerda) e pela implementação paralela com 4 *threads* (direita).

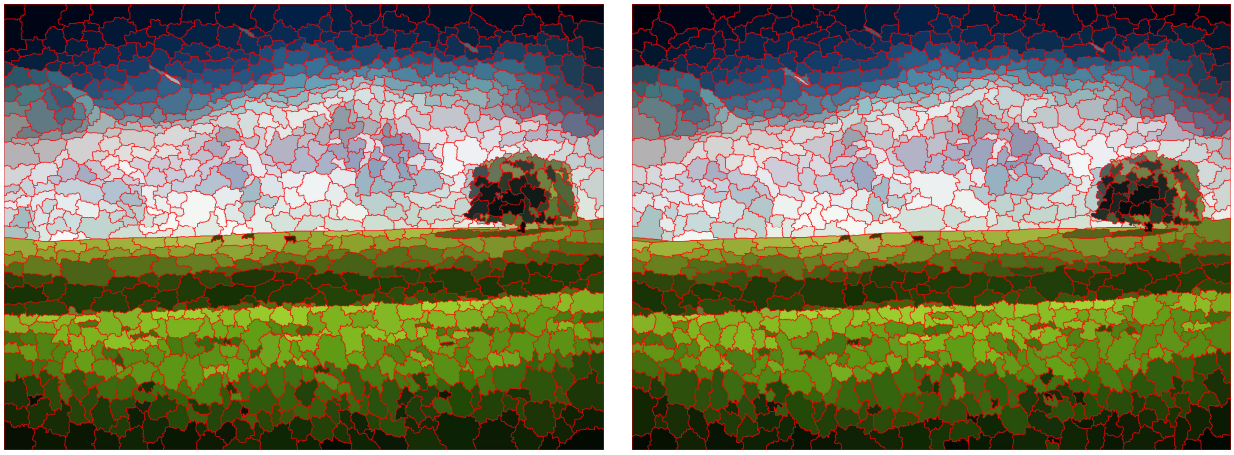

Figura 4. Resultado da segmentação de paisagem pelo algoritmo seqüencial (esquerda) e pela implementação paralela com 4 *threads* (direita).

4.3 Avaliação de Desempenho Computacional

O desempenho computacional do algoritmo paralelo proposto foi avaliado utilizando as mesmas três imagens e variando-se o número de *threads*. A Figura 5 mostra o tempo de execução da segmentação em relação ao número de threads executadas. Nota-se que tempo de execução da segmentação se reduz com o aumento do número de *threads*.

O gráfico sugere uma redução ainda maior do tempo de segmentação se mais *threads* forem utilizadas. Cabe salientar que não compensa executar uma quantidade de *threads* maior do que o número de núcleos do processador. Entretanto, atualmente, muitos *clusters* de alto desempenho apresentam em um nó de processamento, dois ou mais processadores com múltiplos núcleos compartilhando a mesma memória principal. O algoritmo poderia se beneficiar deste tipo de arquitetura, reduzindo ainda mais o tempo de execução.

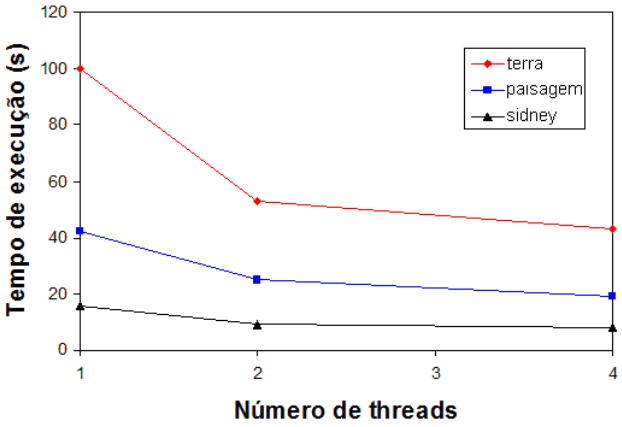

Figura 5. Tempo de execução da implementação paralela para as três diferentes imagens e diferentes números de *threads*.

A Figura 6 mostra a aceleração, ou *speedup*, obtida pelo algoritmo paralelo sugerido. O *speedup* é medido como a razão entre o tempo de execução seqüencial e o tempo de execução paralelo e mostra o aumento relativo de desempenho computacional. Conforme se observa neste gráfico, foram obtidos *speedups* acima de 2 para as imagens *terra* e *paisagem*. Isto decorre do aproveitamento da arquitetura de múltiplos núcleos. O aproveitamento total dos 4 núcleos do processador não foi possível por causa da parte inerentemente seqüencial do algoritmo. Esta parte é necessária para manter a reprodutibilidade dos resultados. A imagem *sidney* foi a que menos se beneficiou do aumento de 2 para 4 *threads* por ser uma imagem pequena.

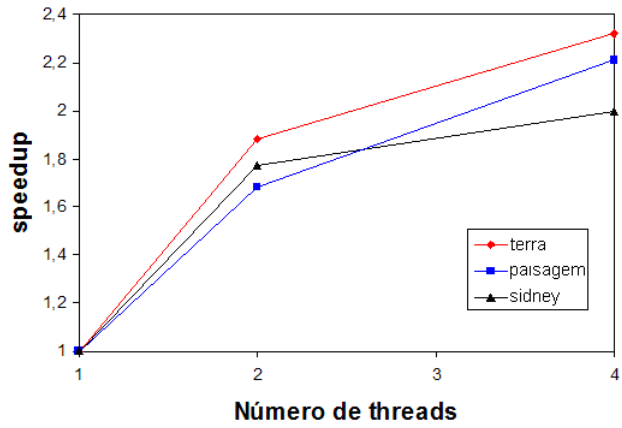

Figura 6. *Speedup* obtido pela implementação paralela para as três diferentes imagens e diferentes números de *threads*.

Convém ressaltar que a implementação do algoritmo de Baatz e Schäpe em que se baseou este trabalho possui uma limitação no que se refere à alocação de memória do sistema. Devido a este fato, não foi possível realizar testes com imagens maiores, nas quais se espera um ganho ainda maior de desempenho computacional.

5. Conclusões

Neste trabalho foi apresentada uma proposta para a paralelização do algoritmo de crescimento de regiões proposta por Baatz e Schäpe. Foi utilizado um algoritmo baseado em

threads com OpenMP de modo a aproveitar a capacidade de processamento paralelo de processadores atuais com múltiplos núcleos.

O foco desta implementação foi melhorar o desempenho da segmentação, mantendo a reprodutibilidade dos resultados. A computação é dividida através de retângulos e o processamento dos segmentos de fronteira é feito seqüencialmente. Em termos de desempenho, a implementação paralela ficou um pouco mais do que duas vezes mais rápida que a segmentação seqüencial. Este é um resultado bastante promissor, porque permite que a grande capacidade de processamento dos processadores atuais com múltiplos núcleos possa ser explorada.

Futuramente, existe a pretensão de utilizar o mesmo princípio da divisão do trabalho em retângulos para escrever uma versão *out-of-core* da segmentação. Esta versão permite a segmentação de imagens que não caibam na memória principal. Dessa forma, espera-se que a segmentação de imagens possa tratar dados extremamente grandes, de forma eficiente e sem a necessidade de hardware especial.

Referências Bibliográficas

Baatz, M.; Schäpe, A. Multiresolution segmentation: an optimization approach for high quality multi-scale image segmentation. In: XII Angewandte Geographische Informationsverarbeitung, Wichmann-Verlag, Heidelberg, 2000.

Chapman, B., Jost G., van der Pas, R. Using OpenMP: Portable Shared Memory Parallel Programming. The MIT Press, 2008.

DEFINIENS, Image Analysis Software for Earth Sciences, http://www.definiens.com/image-analysis-for-earth-sciences_45_7_9.html (último acesso 14 Novembro 2008).