Uma implementação paralela para segmentação de imagens de alta resolução em GPUs

Patrick Nigri Happ [1]
Raul Queiroz Feitosa [1]
Cristiana Bentes [2]
Ricardo Farias [3]

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

[2] Universidade do Estado do Rio de Janeiro - UERJ
Rua São Francisco Xavier, 524, 5º andar - 20550-900 - Rio de Janeiro - RJ, Brasil
cristianabentes@gmail.com

[3] Universidade Federal do Rio de Janeiro - UFRJ
Caixa Postal 68513 - 21945-970 - Rio de Janeiro – RJ, Brasil
rfarias@gmail.com

Abstract. Lately, orbital sensors of high spatial resolution are providing an increasing amount of data about the Earth surface. Analysis of these data implies in a high computational load, which has motivated researches on more efficient hardware and software for these applications. In this context, an issue lies in the image segmentation that involves long processing times and is a key step in object based image analysis. The recent advances in modern programmable graphics units or GPUs have opened the possibility of exploiting the parallel processing capabilities of this hardware to improve the segmentation performance. This work presents a parallel version of the multicriterion segmentation method, introduced originally by Baatz & Schäppe (2000), implemented in a GPU. The underlying hardware architecture consists of a massive parallel and manycore processor system designed for a general purpose computation and especially for image processing. The parallel algorithm is based on processing each pixel as a different thread so as to take advantage of the fine-grain parallel capability of the GPU. In addition to the parallel algorithm, the paper also suggests a modification to the heterogeneity computation that improves the segmentation performance. The experiments under the parallel algorithm show significant results, reaching up to 12 of speedup over the sequential algorithm.

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

1. Introdução

A segmentação de imagens é considerada como etapa fundamental na análise de imagens baseada em objetos geográficos (GEOBIA) (Lübker e Schaab, 2009; Smith e Morton, 2010). Este tópico tem sido tema de diversas pesquisas envolvendo principalmente sua aplicação, otimização de parâmetros, elaboração de novos algoritmos e avaliação de qualidade.

O processo de segmentação de imagens pode envolver um custo de computação elevado (Wassenberg et al., 2009). Este fato torna-se ainda mais evidente ao se considerar o crescente volume de dados decorrente do aumento da resolução e do tamanho das imagens que estão sendo disponibilizadas. Encontrar uma forma de agilizar este método através da computação paralela é uma das possíveis soluções para este problema, como se propõe, por exemplo, em Singh et al. (1999), Lenkiewicz et al. (2009) e Happ et al. (2009).

Unidades de processamento gráfico (GPUs) se tornaram uma ferramenta bastante poderosa para resolverem problemas na área de alto desempenho. As GPUs possuem um elevado grau de processamento e têm superado a capacidade de computação das CPUs. Além disso, GPUs já acompanham grande parte dos computadores pessoais disponíveis no mercado.

O objetivo deste trabalho consiste no desenvolvimento e avaliação de um algoritmo paralelo de segmentação de imagens para ser executado por GPUs. O algoritmo implementado se baseia numa versão de um popular algoritmo de segmentação por crescimento de regiões proposto por Baatz e Schäpe (2000). Algumas alterações são propostas no algoritmo original com a finalidade aproveitar o alto grau de paralelismo disponível na arquitetura da GPU.

O restante deste artigo está organizado da seguinte forma. Na próxima seção a arquitetura básica de uma GPU é descrita brevemente. Na seção 3, o algoritmo original é exposto, assim como a versão paralela proposta neste trabalho. Na seção posterior, a avaliação experimental é detalhada e são expostos os resultados obtidos. Por fim, a seção 5 conclui o trabalho e indica possíveis trabalhos futuros.

2. Arquitetura da GPU

Uma Unidade de Processamento Gráfico ou GPU pode ser vista como um processador de diversos núcleos cuja capacidade computacional provém de uma arquitetura altamente paralela.

O modelo de programação da Nvidia, CUDA, foi criado para o desenvolvimento de aplicações para esta plataforma. CUDA permite que o programador crie funções especiais em C que são executadas em paralelo por *threads* na GPU. As *threads* são organizadas em uma hierarquia de grade de blocos de *threads*. *Threads* de um mesmo bloco podem se sincronizar e compartilhar memória. Durante a execução, os *threads* podem acessar dados em diferentes níveis de hierarquia: registradores, memória compartilhada e memória global. A memória global é acessível por todas as *threads*, mas seu tempo de acesso é em média 500 vezes mais lento que o tempo de acesso à memória compartilhada e aos registradores.

A arquitetura de uma GPU é composta de um conjunto de multiprocessadores ou *streaming multiprocessors*. Cada um destes consiste de um conjunto de *cores*, chamados de *stream processors*. O número de multiprocessadores e de *cores* de uma GPU depende da arquitetura e do modelo da GPU. GPUs modernas podem possuir até 512 *cores*.

A execução das *threads* em uma GPU não é independente. Estas são executados em grupos chamados de *warps*. Dentro de um *warp* todas as *threads* executam a mesma instrução. Quando uma das *threads* diverge das demais no processamento, há uma redução no desempenho, porque esta passa a ser tratada de forma isolada e desabilita o restante das *threads*.

3. Segmentação por Crescimento de Regiões

Entre as técnicas mais comuns para a segmentação de imagens destacam-se aquelas baseadas em crescimento de regiões. Neste método, *pixels* são inicialmente escolhidos e denominados sementes. A partir destes, as regiões crescem através da agregação de outros *pixels* ou sub-regiões adjacentes quando um determinado critério de homogeneidade é atendido. As sementes podem ser selecionadas de maneira aleatória, determinística ou pelo usuário (Pedrini, 2008).

O presente trabalho baseia-se em um algoritmo de crescimento de regiões proposto por Baatz e Schäpe (2000) descrito a seguir.

3.1 Algoritmo Proposto por Baatz & Schäpe

Este algoritmo consiste em um método iterativo que visa minimizar a heterogeneidade média dos objetos da imagem. Todos os *pixels* da imagem são primeiramente considerados como sementes ou segmentos iniciais e a cada passo calcula-se a diferença de heterogeneidade entre cada par de segmentos adjacentes e a homogeneidade do segmento que resultaria da sua fusão. Esta diferença, chamada custo de fusão, deve ser inferior a certo

limiar, chamado escala, para permitir a agregação destes segmentos. O processo se repete até que não seja possível realizar mais nenhuma fusão.

O custo de fusão (*f*) é definido pela soma ponderada de uma componente relativa à heterogeneidade espectral (h_{cor}) e outra relativa à heterogeneidade espacial (h_{forma}). A importância relativa destas componentes é definida por um peso (w_{cor}).

$$f = w_{cor} h_{cor} + (1 - w_{cor}) h_{forma} \quad (1)$$

A heterogeneidade espectral, por sua vez, é dada pela Equação 2, onde Obj_1 refere-se ao segmento selecionado para crescer, Obj_2 a um segmento adjacente e Obj_{12} ao segmento resultante da eventual fusão entre estes. Cada banda da imagem, indexada aqui pelo símbolo *c*, é levada em consideração, sendo multiplicada por um fator w_c que expressa a importância relativa de cada uma. A parcela referente a cada segmento na fórmula é baseada no produto de sua área (*n*) pelo desvio padrão dos valores de seus *pixels* em cada banda (σ_c.)

$$h_{cor} = \sum_c w_c \left(n_{Obj12} \sigma_c^{Obj12} - \left(n_{Obj1} \sigma_c^{Obj1} + n_{Obj2} \sigma_c^{Obj2} \right) \right) \quad (2)$$

A heterogeneidade espacial é constituída por duas componentes de forma: uma referente à compacidade (h_{comp}) e outra relativa à suavidade (h_{svd}). A medida de compacidade *Comp* de um segmento é dada pela razão entre o perímetro *p* e a raiz quadrada da área *n*. Já a suavidade *svd* se refere à razão entre o perímetro do segmento e o perímetro do seu retângulo envolvente mínimo *pbox* (*bounding box*). Como pode ser visto na Equação 3, existe um peso w_{comp} que define o grau de importância relativa entre compacidade e suavidade.

$$h_{forma} = w_{comp} h_{comp} + (1 - w_{comp}) h_{svd} \quad (3)$$

$$Comp = \frac{p}{\sqrt{n}} \quad (4)$$

$$Svd = \frac{p}{\sqrt{pbox}} \quad (5)$$

$$h_{comp} = n_{Obj12} Comp_{Obj12} - \left(n_{Obj1} Comp_{Obj1} + n_{Obj2} Comp_{Obj2} \right) \quad (6)$$

$$h_{svd} = n_{Obj12} Svd_{Obj12} - \left(n_{Obj1} Svd_{Obj1} + n_{Obj2} Svd_{Obj2} \right) \quad (7)$$

O algoritmo possui, portanto, um critério de heterogeneidade ajustável. Parâmetros como a relevância de cada banda espectral e a importância relativa entre forma e cor e entre compacidade e suavidade podem ser regulados com a finalidade de se alcançar uma melhor segmentação. Além disso, a definição do custo máximo de fusão, denominado parâmetro de escala (*s*), influencia diretamente no tamanho dos segmentos gerados.

3.2 Versão Paralela

A versão paralela do algoritmo de segmentação foi desenvolvida nas linguagens C e CUDA da NVidia. A idéia central consiste em associar o processamento de cada *pixel* da imagem a uma *thread* diferente de forma a tirar máximo proveito da alta capacidade de processamento paralelo da GPU.

O algoritmo paralelo consiste em seis funções principais a serem executadas na GPU: Inicializar Sementes, Calcular Vizinhos, Realizar Fusões, Redefinir Segmentos, Recalcular Bordas e Escrever Resultado. A Figura 1 apresenta estas funções e como interagem no algoritmo.

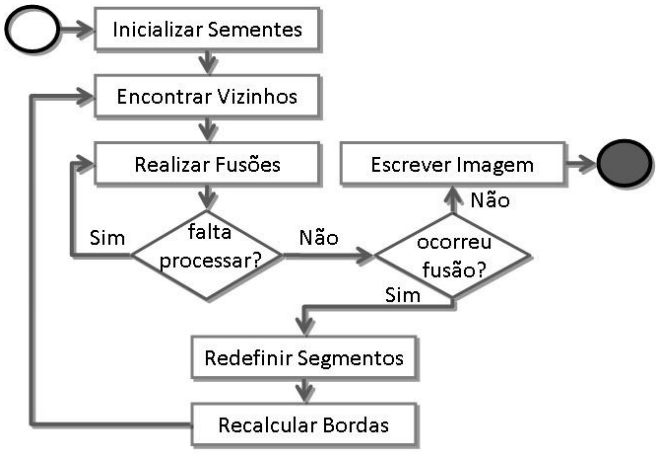

Figura 1. Esquema do algoritmo paralelo de segmentação

O algoritmo começa com a função Inicializar Sementes, onde cada *pixel* é considerado como semente e por conseqüência representa um segmento. Cada um desses *pixels* é processado de forma paralela e tem seus atributos calculados.

A etapa Encontrar Vizinhos é iniciada com um teste que visa restringir o processamento somente aos *pixels* da borda de algum segmento. Para estes *pixels*, são calculados os custos de fusão entre o seu segmento e os segmentos adjacentes. O segmento adjacente que implicar no menor custo é considerado como melhor vizinho. Como o processamento é realizado para todos os *pixels* da borda de um segmento, garante-se, numa seção crítica, que seja identificado o melhor vizinho entre todos vizinhos encontrados para estes *pixels*. .

À medida que os segmentos são aglomerados, apenas um entre os *pixels* de cada segmento é separado para representá-lo. Este passo ocorre na fase Realizar Fusões.

O primeiro passo desta fase consiste na verificação dos *pixels* que representam segmentos, pois somente as suas *threads* devem ser processadas. Após, caso um segmento possua um melhor vizinho que tenha um custo de fusão menor do que o custo de fusão máximo, definido pelo parâmetro de escala, este é selecionado para a fusão.

Uma seção crítica garante que um segmento não seja processado simultaneamente por mais de uma *thread.* Desta forma, os segmentos envolvidos em determinada fusão são marcados, fazendo com que qualquer outra *thread* que tente acessá-los fique bloqueada. O segmento escolhido para fusão engloba seu melhor vizinho e os atributos do segmento resultante são calculados. Por conseqüência, o segmento que foi englobado deixa de ser considerado como segmento e o *pixel* que o representava passa a faze parte do segmento que o anexou.

Cabe ressaltar que, a fim de se evitar perda de desempenho, criou-se um mecanismo para realizar um controle sobre as *threads* que estão em espera. Assim, quando uma *thread* é bloqueada, um indicador é ativado designando que há ainda *pixels* não processados e a *thread* é forçada em seguida a abortar. Ao término desta etapa, caso o indicador tenha sido ativado, este é desativado e a função Realizar Fusões é novamente invocada. Este procedimento se repete até que o indicador não mais seja ativado.

A função Redefinir Segmentos deve ser executada somente se alguma fusão houver ocorrido na etapa Realizar Fusões. Apenas as *threads* dos *pixels* pertencentes a segmentos que foram anexados na função anterior devem ser processadas. Esta função é responsável por atribuir a estes *pixels* o identificador dos segmentos que os englobaram.

A etapa Recalcular Bordas tem o papel de identificar e marcar os *pixels* que deixaram de fazer parte da borda de algum segmento. Desta maneira, para cada *pixel* de borda, é verificado se este está na borda da imagem ou se pelo menos um de seus *pixels* adjacentes pertence a um segmento diferente do seu próprio. Caso contrário, o *pixel* deixa de ser considerado como *pixel* borda.

Quando mais nenhuma fusão é possível, o algoritmo é finalizado com a função Escrever Resultado. As *threads* de cada *pixel* são processadas paralelamente, escrevendo numa imagem de saída, que contém o resultado final da segmentação, a média dos valores espectrais do segmento correspondente.

3.3 Cálculo dos Atributos de Heterogeneidade

O cálculo dos dois atributos de forma descritos na seção 3.1 envolve o cálculo do perímetro (*p*). Para se calcular o perímetro do segmento resultante da fusão de um par de segmentos, é necessário identificar os *pixels* da borda comuns aos dois segmentos. Este procedimento é realizado para cada *pixel* de borda e implica em um grande número de acessos à memória global da GPU, cuja latência é alta. Além disso, este procedimento provoca o desbalanceamento de carga entre as *threads*.

Com o intuito de contornar este problema, é proposta neste trabalho, a substituição destes atributos de forma por outros similares que não envolvam o cálculo do perímetro dos segmentos.

Gomes e Paciornik (2005) apresentam diferentes parâmetros de forma e suas características. Entre estes, destacam-se dois atributos previamente descritos por Russ (1998) que não requerem o cálculo do perímetro: a circularidade e a solidez. O primeiro é um atributo relativo à forma circular dos objetos, assim como a compacidade (Equação 4). Desta maneira, uma variação da circularidade definida pela Equação 8 e denominada de $Atrib_{fcirc}$ foi selecionada para substituir o parâmetro de compacidade no algoritmo proposto. Para esta formulação são levadas em consideração a área *n* e o calibre máximo (*feret* máximo) *c*max dos objetos.

$$Atrib_{fcirc} = \frac{c\max}{\sqrt{n}} \tag{12}$$

A solidez, tal como a suavidade (Equação 5), é um parâmetro que diz respeito à forma convexa dos objetos. Portanto, um atributo definido como $Atrib_{fconv}$,baseado na solidez e definido pela Equação 9, foi escolhido para tomar o lugar da suavidade no novo algoritmo. Para isto, é necessária a utilização da área *n* do objeto e da área de seu retângulo envolvente *nbox*.

$$Atrib_{fconv} = \frac{nbox}{n} \tag{13}$$

4. Avaliação Experimental

Os experimentos descritos nesta seção foram divididos em dois grupos. O primeiro grupo tem como foco a avaliação do impacto da substituição dos atributos de forma previstos no algoritmo original de Baatz & Schäpe pelos novos atributos propostos na seção anterior. O segundo grupo de experimentos visa estimar o ganho de desempenho em função do tempo de processamento da versão paralela em relação à seqüencial.

4.1 Impacto da substituição de atributos de forma

O objetivo destes experimentos é averiguar se a alteração proposta dos atributos de forma é válida, através da análise do resultado obtido por uma determinada função de disparidade.

Cabe salientar que esta é uma verificação preliminar com o intuito apenas de demonstrar que esta substituição não deve penalizar o resultado da segmentação.

Contudo, a relação entre os valores dos parâmetros de segmentação e a qualidade da segmentação resultante não pode ser formulada analiticamente (Feitosa et al., 2009). Desta maneira, foi utilizada uma ferramenta, denominada SPT (*Segmentation Parameters Tuner*), que realiza o ajuste automático dos parâmetros de segmentação através de algoritmo genético.

A imagem empregada nestes experimentos é um recorte de uma imagem QuickBird e sua segmentação foi comparada com três referências contendo, respectivamente, objetos considerados homogêneos, heterogêneos e mesclados. A função de disparidade utilizada foi a RBSB (Costa, 2008) já presente no SPT.

A Tabela 1 exprime o melhor valor obtido pela função de disparidade para cada par de parâmetros de acordo com os tipos de objetos da referência. Pode-se observar que para os três grupos de objetos a diferença em relação aos dois pares de parâmetros é pequena, o que demonstra que o resultado utilizando os novos parâmetros se aproxima do resultado original .

A percepção visual da qualidade da segmentação de cada grupo de objetos em relação a sua respectiva referência pode ser realizada por intermédio da observação da Figura 2. As ilustrações (a), (b) e (c) se referem respectivamente à utilização dos parâmetros de compacidade e suavidade para grupos homogêneos, heterogêneos e mesclados. Já as imagens (d), (e) e (f) são relativas à utilização dos novos parâmetros para objetos homogêneos, heterogêneos e mesclados, respectivamente. As referências são apresentadas na cor amarela.

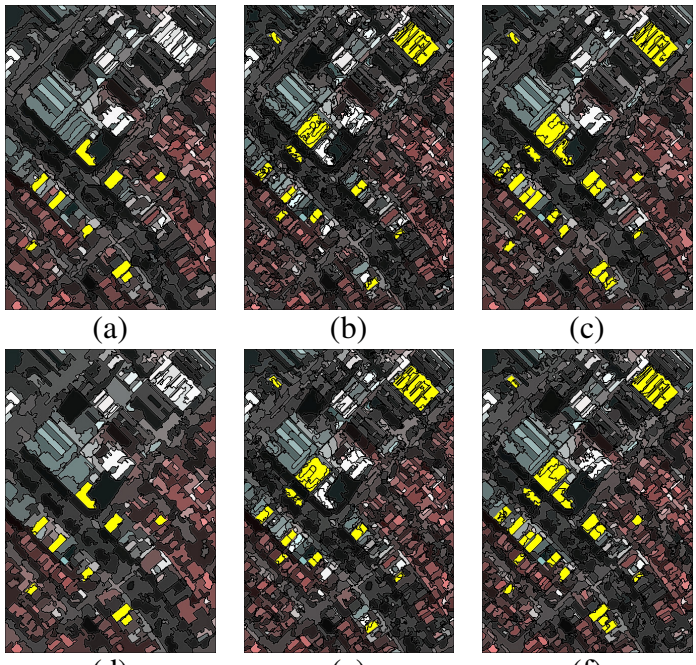

Figura 2. Resultado da segmentação utilizando os parâmetros *Comp* e *Svd* para objetos homogêneos (a), heterogêneos (b) e mesclados (c) e utilizando os parâmetros $Atrib_{fcirc}$ e $Atrib_{fconv}$ também para objetos homogêneos (d), heterogêneos (e) e mesclados (f).

Tabela 1. Valor da disparidade por parâmetros utilizados e tipos de objetos

Parâmetros utilizados	**Valor de disparidade por tipo de objetos**		
	Homogêneos	**Heterogêneos**	**Mesclados**
Comp e *Svc*	0,14	0,56	0,46
$Atrib_{fcirc}$ e $Atrib_{fconv}$	0,16	0,57	0,40

É possível observar a semelhança entre (a) e (d) verificando que, em ambos, as referências estão bem representadas pelos segmentos da imagem. As representações (b) e (e) também são parecidas resultando em segmentos menores que podem ser aglomerados para formar os objetos de referência. O mesmo ocorre para (c) e (f), que assemelham-se entre si. Desta forma, podemos concluir que a alteração dos parâmetros gerou resultados similares aos gerados pelo algoritmo original.

4.2 Aceleração da Versão paralela em Relação à Seqüencial

O objetivo desta seção é avaliar o ganho de desempenho computacional ao se comparar os tempos de execução da versão paralela do algoritmo, executada na GPU, com aqueles da versão seqüencial. Os tempos de execução apresentados mais abaixo se referem à versão modificada do algoritmo. As medidas da versão seqüencial se referem a um computador equipado com processador Core 2 6300 operando a 1.8Ghz e 2GB de RAM. As versões paralelas foram testas em uma GPU Nvidia GeForce 9600 GT e uma GPU Nvidia Tesla. Três diferentes imagens de variados tamanhos (Tabela 2) foram escolhidas para serem processadas.

Tabela 2. Imagens para teste de desempenho e seus respectivos tamanhos

Imagem	Colunas	Linhas
IM1	1024	828
IM2	2800	2800
IM3	3478	2356

A Figura 3 mostra as acelerações (*speedups*) obtidas para cada imagem em cada uma das GPUs em relação à execução seqüencial. Nota-se que, mesmo na GPU de baixo custo (GeForce 9600), alcançou-se um desempenho pelo menos 3 vezes superior ao da versão seqüencial. Para uma GPU com maior poder de processamento, como no caso da Tesla, conseguiu-se uma aceleração de mais de 12 vezes.

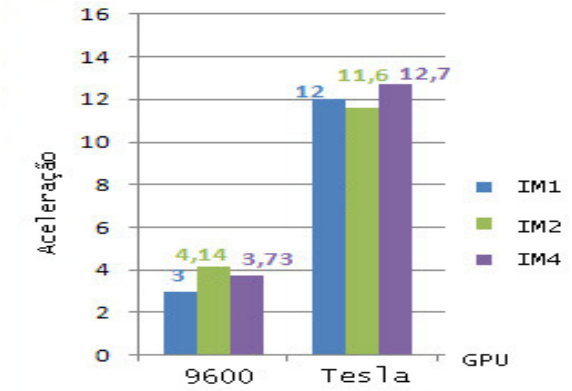

Figura 3. Acelerações obtidas para diferentes imagens e GPUs

5. Conclusões

Este trabalho apresentou um algoritmo paralelo para segmentação de imagens por crescimento de regiões para execução em GPUs. Além da alteração lógica e estrutural do algoritmo, o algoritmo proposto prevê a substituição de parâmetros de forma que caracterizam heterogeneidade espacial, de modo a propiciar um maior ganho de desempenho na versão paralela.

Nos experimentos realizados, a alteração dos atributos proposta neste trabalho não causou alteração substancial no resultado da segmentação, desde que os parâmetros do algoritmo seja

convenientemente ajustados. Testes com maiores massas de dados, se fazem necessários para verificar a generalidade desta afirmativa.

Em relação ao ganho de desempenho, pode-se notar o potencial da utilização da GPU para acelerar processamento da segmentação. A implementação paralela chegou a ser mais do que quatro vezes mais rápida do que a segmentação seqüencial ao se utilizar um *hardware* simples, o que representa um ganho considerável a baixo custo. Para uma GPU mais moderna, foi alcançado um desempenho computacional doze vezes superior ao do algoritmo seqüencial.

A experiência acumulada no desenvolvimento do presente trabalho sugere claramente que o algoritmo aqui proposto não explora todo o potencial das GPUs e que progressos importantes poderão ainda ser alcançados no futuro.

Referências Bibliográficas

Baatz, M.; Schäpe, A. Multiresolution Segmentation: an optimization approach for high quality multiscale image segmentation, Heidelberg. **Strobl J, Blaschke T, Griesebner G (eds) Angewandte Geographische Informationsverarbeitung**, v. 12, p. 12-23, 2000.

Costa, G. A. O. P.; Feitosa, R. Q.; Cazes, T. B.; Feijó, B. Genetic Adaptation of Segmentation Parameters. In: Blaschke, T.; Lang, S.; Hay, G. (Eds.). **Object-Based Image Analysis: Spatial concepts for knowledge-driven remote sensing applications**. Heidelberg: Springer, 2008. p. 679-695.

Feitosa, R. Q.; Costa, G. A. O. P; Fredrich, C. M. B.; Camargo, F. F.; Almeida, C. M. A. Uma avaliação de métodos genéticos para ajuste de parâmetros de segmentação. In: Simpósio Brasileiro de Sensoriamento Remoto (SBSR) 14, 2009. **Anais XIV Simpósio Brasileiro de Sensoriamento Remoto**, Natal, Brasil, 25-30 abril 2009, INPE, p. 6875-6882.

Gomes, O. F. M.; Paciornik, S. Automatic Classification of Graphite in Cast Iron. **Microscopy and Microanalysis**, v. 11, p. 363-371, 2005.

Happ, P. N.; Ferreira, R. S.; Bentes, C.; Costa, G. A. O. P.; Feitosa, R. Q. Segmentação multiresolução: uma abordagem paralela para segmentação de imagens de alta resolução em arquiteturas de múltiplos núcleos. In: Simpósio Brasileiro de Sensoriamento Remoto (SBSR) 14, 2009. **Anais XIV Simpósio Brasileiro de Sensoriamento Remoto**, Natal, Brasil, 25-30 abril 2009, INPE, p. 6935-6942.

Lenkiewicz, P.; Pereira, M.; Freire, M.M.; Fernandes, J. A new 3D image segmentation method for parallel architectures, Multimedia and Expo, 2009. ICME 2009. **IEEE International Conference** on, vol., no., pp.1813-1816, June 28 2009-July 3, 2009.

Lübker, T.; G. Schaab. Optimization of parameter settings for multilevel image segmentation in GEOBIA. In: **Proceedings of ISPRS Hannover Workshop 2009 'High-Resolution Earth Imaging for Geospatial Information'**, 2-5 June 2009, Hannover (Germany), ed. by C. Heipke, K. Jacobsen, S. Müller, and U. Sörgel.

Pedrini, H.; Schwartz, W. **Análise de Imagens Digitais: Princípios, Algoritmos e Aplicações**, Thomson Learning, São Paulo, 2008.

Russ, J. C. **The image processing handbook** - 3rd ed. Materials Science and Engineering Department North Carolina State University Raleigh - North Carolina, 1998.

Singh, D. E.; Heras, D. B.; Rivera, F. F. Parallel seeded region growing algorithm. **Proceedings of VIII Simposium Nacional de Reconocimiento de Formas y Analisis de Imagenes**. SNRFAI'99. Bilbao(Spain), 1999.

Smith, G. M.; Morton, R. D. Real World Objects in GEOBIA through the Exploitation of Existing Digital Cartography and Image Segmentation. **Photogrammetric Engineering & Remote Sensing**, 76 (2). 163-171, 2010.

Wassenberg, J.; Middelmann, W.; Sanders, P. An Efficient Parallel Algorithm for Graph-Based Image Segmentation, CAIP '09: **Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns**, 2009.