UM ALGORITMO DE SEGMENTAÇÃO POR CRESCIMENTO DE REGIÕES PARA GPUS

A region growing segmentation algorithm for GPUs

PATRICK NIGRI HAPP[1]
RAUL QUEIROZ FEITOSA[1,2]
CRISTIANA BENTES[2]
RICARDO FARIAS[3]

[1] Pontifícia Universidade Católica do Rio de Janeiro
Caixa Postal 38097
CEP 22453-900 - Gávea - RJ, Brasil
[2] Universidade do Estado do Rio de Janeiro
Rua São Francisco Xavier, 524
CEP 20550-900 - Maracanã - RJ, Brasil
[3] Universidade Federal do Rio de Janeiro
Caixa Postal 6851
CEP 21945-970, Rio de Janeiro, RJ, Brasil
{patrick, raul}@ele.puc-rio.br; cristianabentes@gmail.com
rfarias@gmail.com

RESUMO

Este artigo propõe um algoritmo paralelo de segmentação de imagens por crescimento de região voltado a Unidades de Processamento Gráfico (GPU). O algoritmo proposto deriva de um algoritmo sequencial largamente utilizado pela comunidade de Análise de Imagens de sensoriamento remoto Baseada em Objeto Geográfico (GEOBIA). Relativamente à versão sequencial propõem-se neste trabalho novos atributos para caracterização de heterogeneidade morfológica de segmentos, cujo cálculo pode ser realizado de modo mais eficiente em GPUs. Duas variantes do algoritmo paralelo com diferentes heurísticas para seleção dos segmentos adjacentes a serem fundidos a cada iteração são descritas. Visando explorar o potencial de GPUs para execução paralela de *threads* de baixa granularidade, o algoritmo proposto atribui uma *thread* para cada pixel da imagem, o que contribui ao mesmo tempo para uma distribuição mais uniforme da carga computacional entre os processadores da GPU. Uma detalhada análise experimental utilizando uma GPU convencional sobre quatro imagens de teste indicou

acelerações superiores a 8 em relação ao algoritmo sequencial.
Palavras-chave: Segmentação paralela; GPU; GEOBIA.

ABSTRACT

This paper proposes a parallel image segmentation algorithm for Graphical Processing Units (GPU) that follows the region-growing paradigm. The proposal can be regarded as a parallel version of a sequential algorithm widely used by the Geographic Object Based Image Analysis community (GEOBIA). In relation to the sequential version this, work presents new attributes to characterize segments' morphological heterogeneity, whose computation can be performed by GPUs more efficiently than the original ones. Two variants of the parallel algorithm with distinct heuristics for the selection of adjacent segments to be merged in each iteration are described. Aiming at exploring the potential of GPUs for the parallel execution of fine grained threads, the new parallel method assigns a thread to each image pixel. This also contributes to better load balance among the GPU processors. A detailed experimental analysis using a simple GPU upon four different test images has shown that the parallel algorithm may run 8 times faster than its sequential counterpart or even more than that.
Keywords: Parallel Segmentation; GPU; GEOBIA.

1. INTRODUÇÃO

Desde o lançamento do satélite IKONOS em 1999 um número crescente de novos sensores orbitais com altíssima resolução espacial (*Very High Resolution – VHR*) para fins não militares entraram em órbita terrestre. A maior resolução das imagens geradas pelos novos sensores orbitais viabilizou um amplo leque de novas aplicações anteriormente reprimidas pelos altos custos das imagens aéreas. Ao mesmo tempo, as perspectivas que se descortinavam trouxeram consigo novos desafios científicos e tecnológicos. Em primeiro lugar, o volume de dados sobre a superfície terrestre disponíveis na forma de imagens digitais cresceu exponencialmente ao longo dos últimos anos, de forma que a análise visual destes dados tornou-se impraticável para muitas aplicações. Por outro lado, os métodos automáticos consagrados para a classificação baseada em pixels, e seus atributos espectrais, mostraram-se inadequados para a interpretação de imagens VHR.

Tornou-se assim premente o desenvolvimento de novos métodos de análise de imagem capazes de explorar além de atributos espectrais outros elementos interpretativos (p.ex., textura, forma ou contexto) aparentes em imagens VHR. Este tema passou então a ser reconhecido como uma nova área de investigação científica denominada Análise de Imagens Baseada em Objeto Geográfico (*Geographic Object Based Image Analysis*), ou simplesmente - GEOBIA.

O primeiro e mais importante passo desta metodologia é segmentação de imagens em regiões homogêneas. Por aproximadamente quatro décadas diferentes algoritmos de segmentação de imagens tem sido propostos (RISEMAN e ARBIB,

1977; FU e MUI, 1981; HARALICK e SHAPIRO, 1985; PAL e PAL, 1993; DEB, 2008). Dentre estes, algoritmos de segmentação por crescimento de regiões têm se destacado especialmente em GEOBIA (TILTON e LAWRENCE, 2000).

Esta técnica de segmentação envolve um alto custo computacional (WASSENBERG et al., 2009) que se torna crítico devido ao volume de dados a ser processado na forma de imagens VHR.

Com o avanço da tecnologia de integração em altíssima escala, organizações paralelas de computadores se tornaram disponíveis comercialmente a preços acessíveis. Assim, a computação paralela se tornou desde há poucos anos a principal alternativa para acelerar o procedimento de segmentação de imagens (MOGA et al., 1998, MONTOYA et al., 2003, LENKIEWICZ et al. 2009, HAPP et al., 2010).

Em particular, as Unidades de Processamento Gráfico (*Graphic Processing Units* - GPUs), que acompanham grande parte dos computadores pessoais disponíveis no mercado, têm se mostrado como uma ferramenta bastante poderosa para resolver algumas classes de problemas que demandam alto desempenho. As GPUs possuem uma elevada capacidade de processamento e superam as CPUs em aplicações que manipulam grandes volumes de dados organizados na forma de vetores e matrizes. Todavia, a programação paralela não é trivial e a maioria das aplicações ainda não se beneficia desse potencial.

Não obstante alguns esforços encontrados na literatura (SCHENKE et al., 2005, BAGGIO, 2007, PAN et al., 2008, XIAO-GU et al, 2009) visando à utilização de GPUs para segmentação de imagens, raros são os trabalho voltados a aplicações em GEOBIA (HAPP et al., 2011, 2012).

O presente trabalho contribui para preencher esta lacuna ao propor um algoritmo paralelo voltado a GPUs para segmentação por crescimento de regiões. Duas versões paralelas do algoritmo são apresentadas baseadas em heurísticas distintas para fusão de segmentos vizinhos.

O algoritmo foi desenvolvido usando as linguagens de programação C e CUDA e seu desempenho computacional foi avaliado em uma GPU de baixo custo e disponível em muitos computadores pessoais à época do estudo.

Cabe mencionar que o presente trabalho dá sequência a estudos sobre segmentação paralela conduzidos pelos autores. Este artigo aprofunda a discussão preliminar apresentada em publicações anteriores (HAPP et al., 2011, 2012), descreve em maiores detalhes o algoritmo paralelo, propõe uma nova e melhor variante do ponto de vista da qualidade final da segmentação, e amplia a análise do desempenho computacional associado.

O restante do artigo está organizado da seguinte forma. A seção 2 descreve sucintamente a arquitetura básica de uma GPU. O algoritmo sequencial de segmentação do qual a presente proposta deriva é introduzido na seção 3. A seção 4 descreve a versão paralela do algoritmo proposto neste trabalho e suas variantes. A avaliação experimental que visou a avaliar o desempenho computacional do algoritmo paralelo proposto é relatada na seção 5. O artigo se encerra sumarizando as principais conclusões do trabalho e indicando direções futuras.

2. ARQUITETURA DA GPU

Inicialmente projetadas para realizar cálculos e tarefas relacionadas ao processamento gráfico em tempo real, as GPUs vêm sendo utilizadas num leque cada vez mais amplo de aplicações.

O modelo de programação da Nvidia para GPUs, CUDA (*Compute Unified Device Architecture*), foi criado visando ao desenvolvimento de aplicações para esta plataforma. CUDA permite que o programador crie funções especiais em C que são executadas paralelamente em diferentes linhas de execução (*threads*) na GPU. As *threads* são organizadas em uma hierarquia definida por uma matriz de blocos, cada um executando um determinado número de *threads*. *Threads* de um mesmo bloco podem operar sincronamente e compartilhar de uma memória específica. Durante a execução, as *threads* podem acessar dados em diferentes níveis da hierarquia de memória: registradores, memória compartilhada e memória global. A memória global é acessível a todas as *threads*, mas seu tempo de acesso é em média 500 vezes maior do que o tempo de acesso à memória compartilhada e aos registradores.

O processamento de programas desenvolvidos em CUDA geralmente se alterna entre a CPU e GPU, que trocam informações (instruções e dados) através da memória principal do computador e da memória interna da GPU. O trecho de código ou função invocado pela CPU para ser executado na GPU é chamado de *kernel*.

A arquitetura de uma GPU é composta de um conjunto de multiprocessadores denominados na literatura do fabricante *streaming multiprocessors*. Cada um destes consiste de um conjunto de núcleos (*cores*), chamados *stream processors*. O número de multiprocessadores e de núcleos de uma GPU depende de sua arquitetura e de seu modelo.

As *threads* são executadas pela GPU em grupos chamados *warps*. Dentro de um *warp* todas as *threads* executam a mesma instrução. Para um bom aproveitamento da capacidade de processamento paralelo das GPUs convém estruturar a aplicação de modo que os *warps* contenham grande número de *threads*, e que a carga de trabalho total seja distribuída o mais uniformemente possível entre estes. Além disso, como o acesso à memória global possui latência elevada, importa que as *threads* tenham baixa granularidade. Em outras palavras, para se maximizar o desempenho os algoritmos a serem executados por GPUs devem ser projetados de modo a consistirem um grande número de *threads* que executam tarefas de baixa complexidade. Maiores informações sobre GPUs da Nvidia, bem como o modelo de programação CUDA podem ser obtidas em (NVIDIA, 2012).

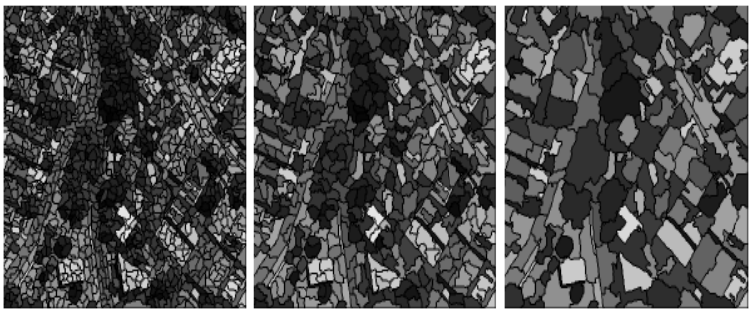
3. SEGMENTAÇÃO POR CRESCIMENTO DE REGIÕES

Os métodos de segmentação baseados em região visam reunir num mesmo conjunto pixels adjacentes que atendem a um dado critério de heterogeneidade. Desta forma, regiões da imagem são agrupadas ou divididas dependendo de seus pixels terem ou não características semelhantes em termos de cor, textura ou forma.

Dentre os métodos desta categoria destacam-se as técnicas de crescimento de regiões (*region growing*), divisão e união de regiões (*split and merge*) e divisor de águas (*watershed*).

O interesse deste trabalho recai sobre os métodos de crescimento de regiões. Nestes, seleciona-se um conjunto inicial de pixels denominados sementes. As regiões crescem a partir das sementes à medida que vão sendo agregadas a pixels ou a sub-regiões vizinhas que atendam a determinado critério de heterogeneidade. As sementes podem ser selecionadas de maneira aleatória, determinística ou pelo usuário (PEDRINI e SCHWARTZ, 2008). A Figura 1 ilustra o processo de crescimento de regiões em três etapas distintas.

Figura 1 – Crescimento de regiões ao longo de iterações.



São dificuldades inerentes a este método a definição do critério de parada, a dependência dos resultados em relação à escolha das sementes e a determinação das características apropriadas para composição do critério de heterogeneidade.

3.1 Algoritmo de Segmentação

Um estudo conduzido por Neubert e coautores (NEUBERT et al., 2008) indicou o algoritmo implementado no software SPRING (CÂMARA et al., 1996) e o algoritmo proposto por Baatz e Schäpe (2000), disponível no software eCognition, como os dois melhores algoritmos de segmentação dentre os mais usados nos anos recentes na área de sensoriamento remoto. Ambos os algoritmos têm em comum o paradigma de "crescimento de regiões" e estão disponíveis como operadores na plataforma InterImage. Diferem, no entanto, quanto ao critério que decide quando dois segmentos adjacentes devem ou não ser mesclados de modo a formarem um único segmento. Enquanto o algoritmo do SPRING considera basicamente apenas atributos espectrais, o algoritmo de Baatz e Schäpe considera tanto atributos espectrais quanto morfológicos.

O algoritmo proposto no presente artigo é uma versão paralela do algoritmo de Baatz e Schäpe. Consiste de um processo iterativo de otimização local, que procura minimizar a heterogeneidade média dos objetos resultantes da imagem.

Inicialmente, todos os pixels da imagem são considerados como sementes ou segmentos iniciais e a cada iteração calcula-se o aumento de heterogeneidade, também chamado custo de fusão, resultante de uma possível fusão entre cada par de segmentos adjacentes existentes. Este valor deve ser inferior a um dado limiar, chamado *escala*, para que a agregação possa ser consumada. O processo se repete até que não seja mais possível realizar fusões.

Cabe esclarecer que para simular um crescimento paralelo das regiões, cada objeto é selecionado apenas uma vez a cada iteração. Além disso, a seleção de objetos é realizada de maneira a escolher objetos relativamente distantes entre si de acordo com a localização na imagem.

O custo de fusão (*f*), representado pela Equação 1, é definido pela soma ponderada de um componente referente à heterogeneidade espectral (h_{cor}) e outro relacionado à heterogeneidade morfológica (h_{forma}). A importância relativa entre os componentes é definida por um peso (w_{cor}) e o cálculo dos componentes se baseia na diferença entre o possível objeto gerado (*obj3*) e a soma independente dos objetos (*obj1* e *obj2*) como mostra a Equação 2.

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

$$h_x = h_{obj3} - (h_{obj1} + h_{obj2}) \quad (2)$$

A heterogeneidade espectral (h_{cor}) é dada pela soma ponderada do desvio padrão dos valores dos pixels que compõe o segmento. Um peso é associado a cada banda espectral que expressa sua importância relativa. A heterogeneidade morfológica (h_{forma}) é composta por dois elementos diferentes: *compacidade* e *suavidade*. A *compacidade* (*Cmp*), formulada na Equação 3, é a razão entre o comprimento de borda (*b*) do segmento e a raiz quadrada de sua área (*n*). Já a *suavidade* (*Svd*) é dada pela razão entre o comprimento de borda (*b*) do segmento e o comprimento de borda (*bbox*) do seu retângulo envolvente mínimo, como define a Equação 4. Um peso definido pelo usuário expressa a importância relativa entre a *compacidade* e a *suavidade* na composição da heterogeneidade morfológica.

$$Cmp = \frac{b}{\sqrt{n}} \quad (3)$$

$$Svd = \frac{b}{bbox} \quad (4)$$

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 ajustados com a finalidade de se alcançar um melhor resultado na segmentação. Como mencionado, há ainda o parâmetro *escala* que define o custo máximo de fusão e influencia diretamente no tamanho dos segmentos gerados.

3.1.1 Heurísticas de decisão para fusão de segmentos

O algoritmo adota uma heurística para eleger para fusão um entre os diversos pares de segmentos que atendem ao critério de heterogeneidade exposto. Quatro alternativas são apresentadas no manuscrito original e descritas a seguir:

1. No procedimento mais simples, denominado neste trabalho de "*vizinho arbitrário*" (*fitting*), dois segmentos adjacentes são fundidos tão logo se determine que atendem ao critério de heterogeneidade. Nesta heurística a seleção dos pares de segmentos agregados é fortemente influenciada pela ordem em que segmentos são processados pelo algoritmo, a chamada sequência de visitação.
2. A heurística do "*melhor ajuste*" (*best fitting*) estabelece que um segmento deve ser fundido com o segmento adjacente que, além de atender ao critério de heterogeneidade, resulte no menor aumento de heterogeneidade entre todos os seus vizinhos. A sequência de visitação tem neste caso menor impacto sobre o resultado da segmentação do que na heurística anterior.
3. Na alternativa denominada "*melhor ajuste mútuo*" (*mutual best fitting*) a fusão se realiza somente se o melhor vizinho (segmento vizinho que resulta no menor aumento de heterogeneidade) de um dado segmento tem neste segmento também o seu melhor vizinho. Ou seja, dado um segmento A, cujo melhor vizinho seja o segmento B, é necessário que A também seja o melhor vizinho do segmento B para que A e B possam ser fundidos num novo e único segmento. Nesta heurística a sequência de visitação tem impacto ainda menor do que a anterior sobre o resultado final da segmentação.
4. A última heurística, denominada "*melhor ajuste global*" (*global mutual best fitting*), determina que somente o par de melhores vizinhos mútuos que resulte no menor aumento de heterogeneidade entre os pares de vizinhos de toda a imagem poderá ser fundido. Nesta heurística o resultado final da segmentação independe da sequência de visitação.

Apesar de apresentar melhor eficiência computacional entre todas as heurísticas aqui consideradas, o "vizinho arbitrário" não é considerado no presente trabalho, pois o resultado da segmentação é muito sensível à sequência de visitação e, por consequência, a pequenas alterações nas dimensões da imagem a ser processada. Não obstante a invariância quanto à sequência de visitação, o "*melhor ajuste global*" é muito oneroso computacionalmente, tendo sido por isso também descartado do presente trabalho.

4. ALGORITMO PARALELO DE SEGMENTAÇÃO

A ideia central do algoritmo proposto consiste em distribuir o processamento relativo a cada pixel da imagem em diferentes *threads*, de forma a tirar o máximo proveito da alta capacidade de processamento da GPU que permite a execução paralela de um número elevado de *threads*. Este tipo de abordagem favorece também um melhor balanceamento de carga, pois cada *thread* sempre se referirá ao tratamento de um único pixel, contribuindo para um processamento de baixa granularidade.

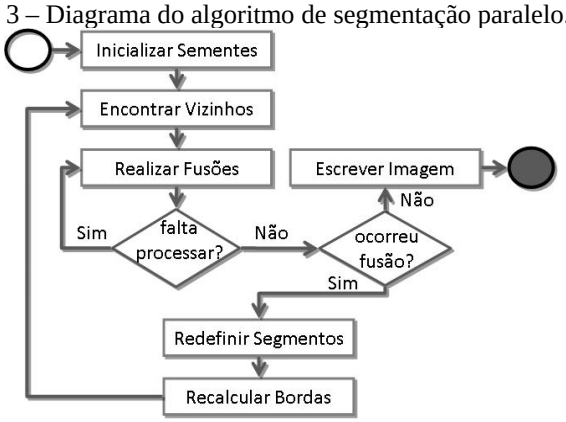
O algoritmo prevê a criação de uma estrutura de dados na memória global da GPU para cada pixel da imagem que armazena informações relativas ao pixel e ao segmento ao qual este pertence. A estrutura de dados consiste de um vetor cujo índice identifica o pixel correspondente.

Cada estrutura contém as seguintes informações: a) o identificador do segmento ao qual o pixel pertence; b) se o pixel pertence ou não à borda deste segmento; c) informações necessárias para o cálculo de atributos espectrais e morfológicos do segmento; d) o identificador do melhor vizinho do segmento; e e) o custo de fusão relativo ao melhor vizinho.

O pixel denominado deste ponto em diante no texto como *formador do segmento* é aquele cujo identificador coincide com o identificador do segmento. As informações relativas aos itens c) até e) só são relevantes para os *formadores do segmento*.

Neste trabalho propõem-se duas variantes do algoritmo paralelo de segmentação. A primeira, denominada "PBF" (*parallel best fitting*), baseia-se no "*melhor ajuste*", e a segunda, chamada "PLMBF" (*parallel local mutual best fitting*), deriva da heurística "*melhor ajuste mútuo*". Ambas as versões são realizadas por seis funções principais (*kernels*) executadas pela GPU (ver Figura 3) e descritas a seguir. Uma descrição mais detalhada está disponível em (HAPP, 2011).

Figura 3 – Diagrama do algoritmo de segmentação paralelo.



1. *Inicializar Sementes*: Esta função inicia de forma paralela cada pixel da imagem como um segmento semente. A função cria ainda a estrutura de dados relativa a cada pixel.
2. *Encontrar Vizinhos:* Esta função possui duas etapas: uma local, referente ao processamento dentro dos blocos de *threads* e uma global para consolidar os resultados locais. Na primeira etapa são identificados os pixels da borda de cada segmento. Calcula-se em seguida para cada pixel de borda o custo de fusão com cada um dos seus segmentos adjacentes. O segmento que apresenta o menor custo de fusão tem seu identificador armazenado localmente na memória compartilhada da GPU relativa àquele bloco. Como o processamento é realizado em paralelo, podem surgir múltiplas indicações de qual seja o melhor vizinho de cada segmento. Tais conflitos são resolvidos no âmbito de cada bloco ao final da primeira etapa. Na segunda etapa, resolvem-se eventuais conflitos entre os diferentes blocos, obtendo-se, assim, para cada segmento uma identificação única de qual é seu melhor vizinho. Esta tarefa é executada em uma seção crítica a fim de atualizar o identificador do melhor vizinho, assim como o custo de fusão referente ao segmento dentro da estrutura do pixel formador deste na memória global da GPU.
3. *Realizar Fusões*: Todos os pixels formadores de segmento com custo de fusão inferior ao valor máximo permitido (*escala*) são selecionados para fusão. Nesta função, uma seção crítica é utilizada para evitar o processamento simultâneo de um mesmo segmento. A fusão propriamente dita é então realizada e a correspondente estrutura de dados é atualizada. Consequentemente, um dos segmentos deixa de existir e seu pixel formador de segmento passa a ser apenas mais um pixel do segmento ao qual foi incorporado. Adicionalmente, esta função rastreia as *threads* ociosas devido à seção crítica fazendo com que sejam abortadas. A função é invocada repetidamente até que nenhuma *thread* seja abortada.
4. *Redefinir Segmentos*: Após a fusão, os pixels de um segmento anexado devem ser identificados como pertencentes ao segmento que os anexou. Para obter um maior desempenho computacional, este procedimento é realizado de forma paralela pela função *Redefinir Segmentos.* Esta função percorre paralelamente estes pixels e atribui-lhes o novo identificador de segmento. A função só é executada no caso da função *Realizar Fusões* ter realizado ao menos uma fusão.
5. *Recalcular Bordas*: Após uma fusão, pixels que antes faziam parte da borda de um segmento podem não pertencer à borda do segmento resultante. Esta função verifica para cada pixel de borda se ao menos um de seus pixels adjacentes pertence a outro segmento. Não sendo este o caso, a correspondente estrutura de dados é atualizada para indicar que tais pixels não mais fazem parte da borda do segmento.

6. *Escrever Imagem:* Quando não é mais possível realizar qualquer fusão, o resultado da segmentação é gerado e o algoritmo é finalizado.

4.2 Diferenças entre as Variantes do Algoritmo

Ambas as variantes, PBF e PLMBF, utilizam as mesmas funções básicas descritas na seção anterior. A diferença fundamental está na função *Realizar Fusões*. Enquanto na PBF basta haver um melhor vizinho que atenda ao critério de heterogeneidade para que ocorra a fusão, na PLMBF requer-se adicionalmente que o segmento que está sendo processado seja também o melhor vizinho de seu melhor vizinho.

Desta forma, após ter selecionado o melhor vizinho, a função *Realizar Fusões* verifica se a condição de mutualidade entre o segmento e seu vizinho é atendida. Em caso negativo, não se realiza a fusão.

A heurística de decisão do "*melhor ajuste mútuo*" pode produzir resultados anômalos quando mais de um par de candidatos à fusão apresentam igual custo de fusão. Esta situação ocorre com relativa frequência quando se opera sobre regiões homogêneas da imagem.

O algoritmo proposto incorpora um mecanismo que lida com tais situações armazenando para cada segmento, os vários melhores vizinhos que apresentam o mesmo custo de fusão. Além da modificação na estrutura de dados, introduzem-se alterações nas funções *Encontrar Vizinhos* e *Realizar Fusões*. Maiores detalhes podem ser encontrados em (HAPP, 2011). Cabe mencionar que esta mudança traz consigo um ganho de desempenho tanto na versão paralela quanto na sequencial do algoritmo.

4.3 Precisão da GPU para Operações em Ponto Flutuante

De um modo geral, a GPU tem precisão inferior à CPU em operações de ponto flutuante. Como consequência, ocorre com mais frequência na GPU do que na CPU que o cálculo de uma expressão aritmética em ordens distintas, mas algebricamente equivalentes, produza resultados diferentes devido a erros de arredondamento. Tais erros podem ter impactos substanciais no resultado final do algoritmo, dependendo da precisão com que se opera. Tendo em vista que a origem de tais problemas pode ser de difícil identificação para neófitos no uso de GPUs, considerou-se adequado descrevê-la nesta seção, bem como apresentar a solução concebida neste trabalho.

No caso da versão PLMBF, o problema em tela pode provocar a super segmentação de algumas áreas da imagem. Em alguns casos, ao se calcular o custo de fusão do segmento A com o segmento B, o valor obtido pode ser diferente do custo calculado para a fusão do segmento B com o segmento A. Estas diferenças podem levar a que pares de "melhores vizinhos mútuos" não sejam identificados como tais, dando origem a segmentos que não mais se fundem com nenhum de seus vizinhos, gerando por fim segmentos pequenos, ou seja, super segmentação.

O problema pode ser solucionado através de uma heurística simples. Basicamente, antes de realizar o cálculo de uma expressão algébrica, verifica-se

inicialmente qual dos segmentos possui o menor identificador. O cálculo dos atributos é feito sempre deste em relação ao seu vizinho. Garante-se, assim, uma ordem fixa no cálculo de expressões aritméticas, e o problema deixa de se apresentar.

4.4 Cálculos dos Atributos de Heterogeneidade

Os atributos definidos nas Equações 3 e 4 devem ser calculados sempre que se cogita de fundir dois segmentos adjacentes. Neste caso, há que se calcular o comprimento da borda do novo segmento resultante da fusão. Para tanto, somam-se os pixels de borda de cada um dos dois segmentos adjacentes considerados para fusão e se subtraem os pixels da borda comum entre os segmentos. Entretanto, este é um procedimento bastante caro computacionalmente e que requer ainda um grande número de acessos à estrutura dos segmentos presente na memória global da GPU, cuja latência é alta. Além disso, também provoca o desbalanceamento de carga entre as *threads*, pois o processamento é proporcional ao tamanho da borda de cada segmento.

Com o intuito de contornar este problema, propõe-se a substituição da *compacidade* e da *suavidade* por outros atributos cujo cálculo não envolve a determinação do comprimento da borda dos segmentos e que alcançam seus valores mínimos para as mesmas formas típicas, respectivamente, círculo e retângulo.

Inspirados na discussão apresentada por Russ (1998) sobre atributos morfológicos, propõe-se neste trabalho uma nova medida para *compacidade* (*Comp*) definida pela Equação 5,

$$Comp = \frac{d\max}{\sqrt{\frac{4n}{\pi}}} \tag{5}$$

onde *n* representa a área do objeto e *dmax* o eixo da elipse que apresenta momento de segunda ordem idêntico ao do segmento. Cabe notar que o valor mínimo é alcançado por segmentos circulares, assim como a definição de compacidade dada pela Equação 3.

Propõe-se ainda a substituição da *suavidade* pelo atributo *solidez* (*Sol*) definido pela Equação 6.

$$Sol = \frac{nbox}{n} \tag{6}$$

A solidez varia em função da convexidade do segmento e seu cálculo envolve a área *n* do segmento e a área de seu retângulo envolvente mínimo *nbox*.

5. ANÁLISE EXPERIMENTAL

5.1 Impacto da Substituição dos Atributos de Forma sobre o Resultado da Segmentação

A hipótese subjacente à modificação sugerida na seção 4.4 é de que com os atributos definidos pelas Equações 5 e 6 será sempre possível chegar a uma segmentação similar à produzida pelo algoritmo de Baatz e Schäpe (2000) em sua formulação original, mediante um adequado ajuste dos parâmetros do algoritmo.

A demonstração de que tal hipótese tem validade geral constitui um desafio que excede as pretensões do presente trabalho. Nesta seção limitam-se os autores a testá-la experimentalmente em algumas amostras de imagens.

Para este propósito, selecionou-se um recorte de uma imagem QuickBird. Três conjuntos de segmentos, chamados segmentos de referência, foram delineados manualmente por um foto-intérprete a fim de representarem a segmentação ideal para três diferentes classes de objetos nesta imagem.

Um procedimento similar ao descrito em (FEITOSA et al., 2006) foi aplicado para determinar o conjunto de valores dos parâmetros de segmentação que produz o resultado mais consistente com cada conjunto de referências. Consistência neste caso corresponde ao menor valor de uma medida de discrepância entre referências e resultado da segmentação. Nestes experimentos, foi utilizada a medida de discrepância denominada *Reference Bounded Segments Booster* (RBSB) (FEITOSA et al., 2006). Este procedimento foi realizado utilizando-se o algoritmo sequencial com "*melhor ajuste*" e variando os atributos de forma: originais e propostos.

A Tabela 1 apresenta o melhor valor da medida de discrepância obtido para cada par de atributos para os três grupos de referências. As diferenças nos valores de discrepância ao longo de uma mesma coluna da Tabela 1 correspondem visualmente a pequenas diferenças da qualidade da segmentação (HAPP et al., 2012).

Muito embora não constituam prova cabal, estes resultados corroboram a expectativa de que a substituição dos atributos de forma, conforme se propõe neste artigo, preserva o comportamento do algoritmo original, desde que se proceda ao adequado ajuste dos parâmetros de segmentação.

Tabela 1 - Valor de discrepância por atributos utilizados e conjuntos de referência.

	Valor de discrepância por grupos de referência		
Atributos	**Grupo 1**	**Grupo 2**	**Grupo 3**
Cmp & Svd	0.14	0.56	0.46
Comp & Sol	0.16	0.57	0.40

5.3 Análises do Desempenho Computacional

As linhas que se seguem apresentam a análise experimental realizada com o propósito de avaliar o desempenho computacional das propostas trazidas no presente trabalho.

O primeiro grupo de experimentos visou avaliar o ganho computacional decorrente da substituição dos atributos morfológicos. No segundo grupo de experimentos mediu-se a aceleração alcançada pela versão paralela em relação à versão sequencial do algoritmo.

Em ambos os grupos foram utilizadas quatro imagens diferentes denominadas a seguir *Quadra, Serra, Cidade*, e *Maracanã*, cujos tamanhos são indicados na tabela 2. *Quadra* é um recorte de uma imagem QuickBird. *Serra* e *Cidade* são recortes de imagens IKONOS e *Maracanã* é uma imagem aérea analógica pancromática digitalizada a 300 DPI. Cabe mencionar que para efeitos de teste foram consideradas apenas três bandas espectrais.

Tabela 2 - Imagens de teste.

Rótulo	Colunas	Linhas	Total de Pixels	Sensor
Quadra	1000	1000	1.000.000	QuickBird
Serra	1600	1600	2.560.000	IKONOS
Cidade	2311	2086	4.820.746	IKONOS
Maracanã	2732	2456	6.709.792	Imagem Aérea

Com exceção da *escala*, todos os demais parâmetros do algoritmo foram mantidos fixos em todos os experimentos.

5.3.1 Desempenho das versões sequenciais após substituição dos atributos

Esta seção apresenta os experimentos conduzidos com o propósito de avaliar a aceleração no algoritmo sequencial obtida pela alteração dos atributos espaciais.

A Figura 7 apresenta o gráfico de aceleração em função da *escala* para a heurística de "*melhor ajuste*". A linha horizontal marca a aceleração unitária. Nota-se que a aceleração é inferior a 1 (um) para *escala*s até 20. Conclui-se que a alteração dos atributos de forma proposta neste trabalho pode ser ligeiramente deletéria para os tempos de processamento quando se trabalha com *escala*s muito baixas. Ressalve-se, contudo, que raramente se utiliza *escala* nesta faixa em aplicações práticas.

Por outro lado, observa-se no gráfico uma tendência de aumento da aceleração com o aumento da *escala*. De fato, os valores observados estiveram acima de 1 (um) na quase totalidade dos experimentos realizados quando se operou com *escala*s superiores a 20.

Em resumo, a aceleração mínima obtida foi de 0,93 e a máxima de 1,1, o que leva à conclusão de que a substituição proposta dos atributos de forma não traz consigo alteração significativa nos tempos de processamento da versão sequencial.

A Figura 8 apresenta os gráficos de aceleração para a heurística de "*melhor ajuste mútuo*". Novamente foi traçada no gráfico uma linha referente à aceleração de valor 1. Neste caso, verifica-se que a aceleração das imagens *Serra* e *Maracanã* encontram-se abaixo desse limite para as *escala*s 5 e 10, enquanto para a imagem

Cidade apenas na *escala* 5. Para a imagem *Quadra* só se obtiveram acelerações acima de um. De qualquer forma, como já se enfatizou, *escala*s tão baixas são raramente utilizadas na prática.

De um modo geral, a partir da *escala* igual a 20, a aceleração foi superior a 1 (um) e cresceu com a *escala* em todos os experimentos. No quadro geral, a menor aceleração registrada foi próxima a 0,93 para *Cidade* e a maior foi de 1,57, também para *Cidade*.

Figura 7 – Gráfico de aceleração em função da *escala* para o algoritmo sequencial operando com a heurística de "*melhor ajuste*" e novos atributos de forma.

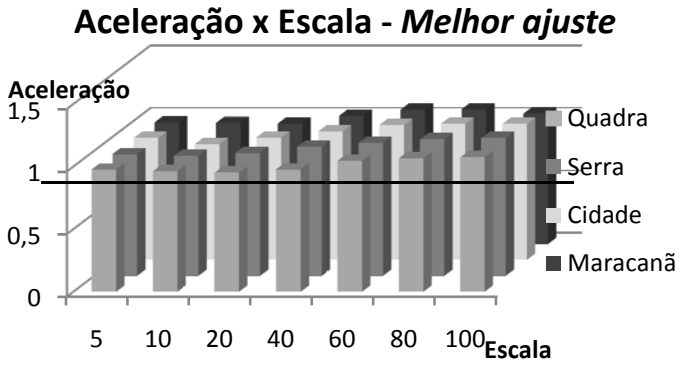
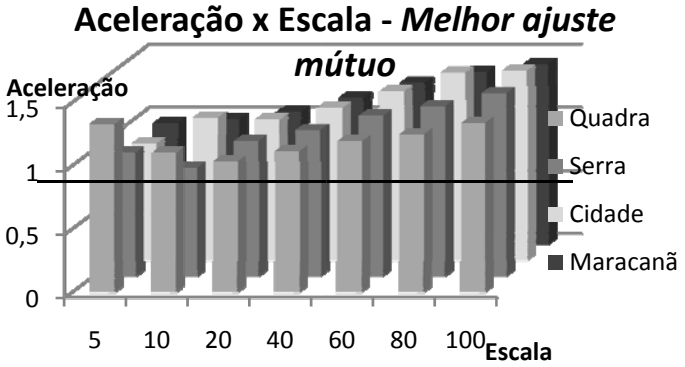


Figura 8 – Gráfico de aceleração em função da *escala* para o algoritmo sequencial operando com a heurística de "*melhor ajuste mútuo*" e novos atributos de forma.



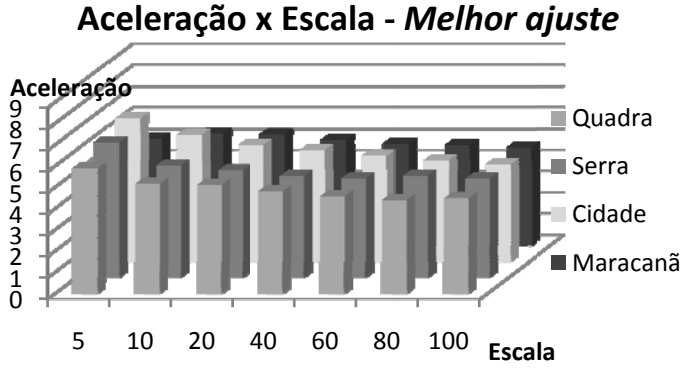
5.3.2 Desempenho das versões paralelas em relação às sequenciais

O objetivo da série de experimentos relatados a seguir foi determinar a aceleração alcançada pelas versões paralelas do algoritmo de segmentação propostas neste artigo. Os experimentos foram executados em uma configuração de hardware básica e de baixo custo. A GPU NVIDIA GeForce 9600 GT com 64 núcleos e 1GB de memória foi utilizada para o processamento paralelo. O Sistema Operacional em execução foi o Windows XP e o processador um *Core 2* 6300 operando a 1.86Ghz com a disponibilidade de 3.25GB de memória RAM.

A Figura 9 exibe o gráfico de aceleração para diversos valores de *escala* do algoritmo paralelo de "*melhor ajuste"* em relação à sua versão sequencial.

À medida que a *escala* aumenta nota-se uma queda da aceleração. Para imagem *Quadra*, a aceleração variou entre 5,89 e 4,49, para *Serra* entre 6,37 e 4,65, para *Cidade* entre 6,74 e 4,58 e, finalmente, para *Maracanã* entre 5,29 e 4,63.

Figura 9 – Gráfico de aceleração em função da *escala* para o algoritmo paralelo de "*melhor ajuste*" em relação à respectiva versão sequencial.



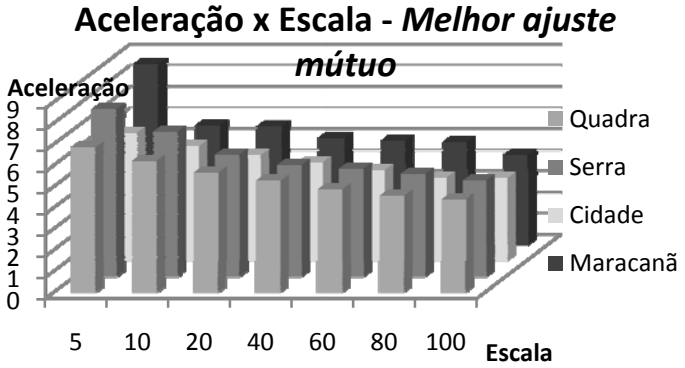
A Figura 10 mostra o gráfico de aceleração para diferentes valores de *escala* para o algoritmo paralelo de "*melhor ajuste mútuo*". Para a imagem *Quadra*, a aceleração da versão paralela variou entre 6,86 e 4,42. Para *Serra* a aceleração ficou entre 7,93 e 4,56, para *Cidade* entre 7,93 e 3,91, e para *Maracanã* entre 8,57 e 4,27. Neste caso, nota-se que a aceleração é muito mais sensível à imagem de entrada do que no caso da versão paralela do "*melhor ajuste*".

A explicação para o decréscimo da aceleração em relação ao aumento da *escala* está relacionada ao número de iterações que ambos os algoritmos necessitam para concluir a tarefa. O processamento dentro de cada iteração diminui à medida que as iterações se sucedem. Assim, os custos decorrentes de sincronizações e de acessos à memória global passam a ser cada vez mais significativos quando comparados ao custo de processamento. Além disso, havendo menos processamento, o alto grau de paralelismo da GPU é menos aproveitado, deixando

muitas *threads* ociosas. Portanto, os benefícios da GPU se tornam mais evidentes para escalas menores uma vez que implicam em um menor número total de iterações.

De um modo geral, a razão entre custo de processamento e sincronizações/acessos à memória é mais elevada na versão "*melhor ajuste mútuo*" do que na "*melhor ajuste*". Desta forma, as acelerações observadas em nossos experimentos foram melhores para o "*melhor ajuste mútuo*". Essa superioridade se torna mais clara para *escalas* menores.

Figura 10 – Gráfico de aceleração em função da *escala* para o algoritmo paralelo de "*melhor ajuste mútuo*" em relação à respectiva versão sequencial.



Cabe ainda ressaltar que apesar de alcançar acelerações de até 8,57 vezes em relação à versão sequencial, estas ficaram bem abaixo do número de núcleos utilizados na GPU. Isto ocorre devido às características do algoritmo de segmentação que possui uma forte dependência entre threads e envolve acessos frequentes à memória global.

6. CONCLUSÃO

Propôs-se neste trabalho um algoritmo paralelo de segmentação de imagens em GPUs. Mais especificamente, foram apresentadas e avaliadas neste trabalho duas variantes paralelas de um algoritmo de segmentação amplamente utilizado pela comunidade de sensoriamento remoto em que o critério de heterogeneidade que controla o crescimento de regiões é formulado tanto em termos de atributos espectrais quanto morfológicos.

Cada variante é baseada numa heurística de decisão diferente para fusão de segmentos. A primeira é fundamentada na fusão do "*melhor ajuste*" e a segunda na fusão do "*melhor ajuste mútuo*". Em ambas as variantes, o algoritmo paralelo apresentado trata cada pixel em uma *thread* distinta, visando explorar ao máximo a capacidade de processamento paralelo oferecida pela GPU. Com objetivo

semelhante, propôs-se a substituição dos atributos morfológicos que compõem o critério de heterogeneidade no algoritmo sequencial original por novos atributos, cujo cálculo pode ser realizado de modo mais eficiente pelas GPUs.

Verificou-se experimentalmente que os novos atributos morfológicos não implicam em alteração significativa nos resultados da segmentação, desde que os parâmetros do algoritmo sejam adequadamente ajustados.

A análise experimental estimou ainda o potencial da utilização da GPU para acelerar este tipo de aplicação. Utilizando um hardware simples (GeForce 9600 GT) foi possível atingir valores de aceleração de até 8,57.

Concluindo, cumpre ressaltar em primeiro lugar que a segmentação é responsável por parcela importante do tempo de processamento em diversas aplicações, em particular Análise de Imagens Baseada em Objeto Geográfico. Em segundo lugar, os ganhos de desempenho computacional medidos experimentalmente neste estudo foram alcançados com GPUs de baixíssimo custo presentes em configurações básicas da maioria dos computadores pessoais. A solução proposta apresenta, portanto, uma relação custo/benefício muito atraente para aplicações em GEOBIA.

É ainda razoável supor que ganhos de desempenho ainda maiores sejam alcançados no futuro à medida que GPUs cada vez mais velozes se tornem disponível.

Uma implementação do algoritmo proposto nas duas variantes, bem como suas versões sequenciais, estão disponíveis exclusivamente para fins educacionais no seguinte endereço: http://www.lvc.ele.puc-rio.br/wp/?cat=41.

AGRADECIMENTOS:

Este estudo é consequente da dissertação de mestrado do primeiro autor. Ele gostaria de expressar seu agradecimento ao CNPq pelo apoio financeiro, fundamental para o desenvolvimento deste trabalho.

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.

BAGGIO, D. L. *GPGPU Based Image Segmentation Livewire Algorithm Implementation*. Master's Thesis. Aeronautical Institute of Technology, São José dos Campos, 2007.

CÂMARA, G.; SOUZA, R. C. M.; FREITAS, U. M.; GARRIDO, J.; II, F. M. SPRING: Integrating remote sensing and GIS by object-oriented data modeling. *Computers & Graphics*, 20: (3) 395-403, May-Jun 1996.

DEB, S. Overview of image segmentation techniques and searching for future directions of research in content-based image retrieval, *Ubi-Media Computing, 2008 First IEEE International Conference on* , vol., no., pp.184-189, 2008.

FEITOSA, R. Q.; COSTA, G. A. O. P..; CAZES, T. B.; FEIJÓ, B. A genetic approach for the automatic adaptation of segmentation parameters. International Conference on Object-based Image Analysis – ISPRS *Proceedings*, v. 36, n. 4/C42, 2006.

FU, K. S.; MUI, J. K. A survey on image segmentation, *Pattern Recognition* 13 (1981), pp. 3–16.

HAPP, P. N., FERREIRA, R. S., BENTES, C., COSTA, G. A. O. P,; FEITOSA, R. Q. Multiresolution Segmentation: a Parallel Approach for High Resolution Image Segmentation in Multicore Architectures, In: 3rd International Conference on Geographic Object-Based Image Analysis, 2010, Ghent, *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Enshede: ITC, 2010. v.XXXVII.

HAPP, P. N. *Segmentação de imagens em GPUs: uma abordagem paralela para crescimento de regiões*. Dissertação de Mestrado. Rio de Janeiro, Brasil: Pontifícia Universidade Católica do Rio de Janeiro, 2011.

HAPP, P. N.; FEITOSA, R. Q; BENTES, C.; FARIAS, R. Q. Uma implementação paralela para segmentação de imagens de alta resolução em GPUs. In: Simpósio Brasileiro de Sensoriamento, 15. (SBSR), 2011, Curitiba. *Anais...* São José dos Campos: INPE, 2011. p. 7713-7720.

HAPP, P. N.; FEITOSA, R. Q; BENTES, C.; FARIAS, R. Q. A Parallel Image Segmentation Algorithm on GPUS. In: International Conference on Geographic Object-Based Image Analysis, 4. (GEOBIA), 2012, Rio de Janeiro. *Proceedings...*São José dos Campos: INPE, 2012. p. 580-585.

HARALICK, R. M.; SHAPIRO, L. G. Image segmentation techniques. *Comput. Vision Graphics Image Process*., 29:100–132, 1985.

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.

MOGA, A.; CRAMARIUC, B.; GABBOUJ, M. Parallel watershed transformation algorithms for image segmentation, *Parallel Computing*, 1998.

MONTOYA, M. D. G.; GIL, C.; GARCÍA, I. The load unbalancing problem for region growing image segmentation algorithms, *Journal of Parallel and Distributed Computing*, 2003.

NEUBERT, M.; HEROLD, H.; MEINEL, G.. Assessment of Remote Sensing Image Segmentation Quality. Proceedings GEOBIA 2008, Calgary, Canada, August, 6-7, 2008, *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XXXVIII-4/C1, 5 p.

NVIDIA.CUDA C *ProgrammingGuide*,v5.0, Disponível em: <http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf > Acesso: novembro, 2012.

PAL, N. R.; PAL, S. K. A review of image segmentation techniques, *Pattern Recognition*, 26(9):1277-94, 1993.

PAN, L.; GU, L.; XU, J. Implementation of medical image segmentation in CUDA. *Information Technology and Applications in Biomedicine*, 2008. ITAB 2008. International Conference on , vol., no., pp.82-85, 30-31 maio 2008.

PEDRINI, H., SCHWARTZ, W. *Análise de Imagens Digitais: Princípios, Algoritmos e Aplicações*, Thomson Learning, São Paulo, 2008.

RISEMAN, E. M.; ARBIB, M. A. Computational techniques in the visual segmentation of static scenes, *Comput. Vision Graphics Image Process.* 6 (1977), pp. 221–276.

RUSS, J. C.. *The Image Processing Handbook - 3rd ed*. Materials Science and Engineering Department North Carolina State University Raleigh - North Carolina, 1998.

SCHENKE S., WUENSCHE B., DENZLER J. GPU-based volume segmentation. In *Proeeidings.* of IVCNZ '05 (2005), pp. 171 – 176.

XIAO-GU, SUN; MAN-CHUN, LI; YONG-XUE, LIU; WEI, LIU; LU, TAN. Accelerated segmentation approach with cuda for high spatial resolution remotely sensed imagery based on improved mean shift. In *Urban Remote Sensing Joint Event*, pages 1–6.

TILTON, J. C.; LAWRENCE, W. T. Interactive analysis of hierarchical image segmentation, Geoscience and Remote Sensing Symposium, 2000. *Proceedings.* IGARSS 2000. IEEE 2000 International, vol.2, no., pp.733-735.

WASSENBERG, J., MIDDELMANNAND, W.; SANDERS, P. An efficient parallel algorithm for graph-based image segmentation. In CAIP '09: *Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns*, pp. 1003-1010, Berlin, Heidelberg. Springer-Verlag.

(Recebido em novembro de 2012. Aceito em março de 2013).