Pontifícia Universidade Católica
do Rio de Janeiro

Patrick Nigri Happ

Segmentação de Imagens Distribuída Baseada em MapReduce

Tese de doutorado

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da PUC-Rio como requisito parcial para obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Prof. Raul Queiroz Feitosa
Co-Orientador: Prof. Gilson Alexandre Ostwald Pedro da Costa

Rio de Janeiro
Agosto de 2015

Patrick Nigri Happ

Segmentação de Imagens Distribuída Baseada em MapReduce

Tese apresentada como requisito parcial para obtenção do título de Doutor pelo Programa de Pós-Graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Raul Queiroz Feitosa
Orientador
Departamento de Engenharia Elétrica - PUC-Rio

Prof. Gilson Alexandre Ostwald Pedro da Costa
Co-Orientador
Departamento de Engenharia Elétrica - PUC-Rio

Prof. Bruno Feijo
Departamento de Informática - PUC-Rio

Prof. Sidnei Paciornik
Departamento de Engenharia Química e de Materiais - PUC-Rio

Prof.ª Cristiana Bentes
UERJ

Prof. Ricardo Farias
UFRJ

Prof. José Eugênio Leal
Coordenador Setorial do Centro
Técnico Científico

Rio de Janeiro, 27 de agosto de 2015

Patrick Nigri Happ

Possui graduação em Engenharia Elétrica com ênfase em Sistemas e Computação pela Universidade do Estado do Rio de Janeiro (2008) e mestrado em Engenharia Elétrica pela Pontifícia Universidade Católica do Rio de Janeiro (2012). Desenvolve pesquisas na área de análise de imagens, sensoriamento remoto, processamento paralelo e distribuído.

Ficha Catalográfica

Happ, Patrick Nigri

Segmentação de imagens distribuída baseada em MapReduce / Patrick Nigri Happ ; orientador: Raul Queiroz Feitosa ; co-orientador: Gilson Alexandre Ostwald Pedro da Costa. – 2015.

104 f. : il. (color.) ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica, 2015.

Inclui bibliografia

1. Engenharia elétrica – Teses. 2. Sensoriamento Remoto. 3. Análise de imagens. 4. Segmentação de imagens. 5. Computação em Nuvem. 6. Processamento distribuído. I. Feitosa, Raul Queiroz. II. Costa, Gilson Alexandre Ostwald Pedro da. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. IV. Título.

CDD: 621.3

Para Aline e minha família.

Agradecimentos

Agradeço a todos que de forma direta ou indireta auxiliaram na execução desta Tese e em especial:

Aos meus orientadores, Prof. Raul Queiroz Feitosa e Prof. Gilson Alexandre Ostwald Pedro da Costa, pelo apoio, dedicação e contribuições durante a pesquisa de Doutorado.

Aos amigos e colegas do Laboratório de Visão Computacional, principalmente Rodrigo Ferreira, Dario Oliveira e Victor A. Quirita que participaram da pesquisa e desenvolvimento do InterIMAGE Cloud Platform, que muito contribuiu para o desenvolvimento desta tese.

Ao Prof. Paolo Gamba por me receber na Universidade de Pavia assim como aos colegas e amigos desta acolhedora cidade que tornaram minha estada ainda mais produtiva.

Ao CNPq, à CAPES e à PUC-Rio pelos auxílios financeiros prestados fundamentais para o desenvolvimento deste trabalho.

À minha mãe Faride (*in memoriam*) por ser fonte de inspiração e por ter me tornado quem sou.

Aos meus grandes amigos e à minha família, o que inclui aqueles que não necessariamente são do mesmo sangue, mas que sempre demonstraram o mesmo apoio e afeto.

À minha esposa Aline por toda ajuda, paciência e compreensão.

Resumo

Happ, Patrick Nigri; Feitosa, Raul Queiroz; Costa, Gilson Alexandre Ostwald Pedro da. **Segmentação de imagens distribuída baseada em MapReduce**. Rio de Janeiro, 2015. 104p. Tese de Doutorado - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

A Segmentação de imagens representa uma etapa fundamental na análise de imagens e geralmente envolve um alto custo computacional, especialmente ao lidar com grandes volumes de dados. Devido ao significativo aumento nas resoluções espaciais, espectrais e temporais das imagens de sensoriamento remoto nos últimos anos, as soluções sequenciais e paralelas atualmente empregadas não conseguem alcançar os níveis de desempenho e escalabilidade esperados. Este trabalho propõe um método de segmentação de imagens distribuída capaz de lidar, de forma escalável e eficiente, com imagens grandes de altíssima resolução. A solução proposta é baseada no modelo MapReduce, que oferece uma estrutura altamente escalável e confiável para armazenar e processar dados muito grandes em ambientes de computação em *clusters* e, em particular, também para nuvens privadas e comerciais. O método proposto é extensível a qualquer algoritmo de crescimento de regiões podendo também ser adaptado para outros modelos. A solução foi implementada e validada usando a plataforma Hadoop. Os resultados experimentais comprovam a viabilidade de realizar a segmentação distribuída sobre o modelo MapReduce por intermédio da computação na nuvem.

Palavras-chave

Sensoriamento Remoto; Análise de Imagens; Segmentação de Imagens; Computação em Nuvem; Processamento Distribuído.

Abstract

Happ, Patrick Nigri; Feitosa, Raul Queiroz (Advisor); Costa, Gilson Alexandre Ostwald Pedro da (Co-Advisor). **A distributed region growing image segmentation based on MapReduce.** Rio de Janeiro, 2015. 104p. Ph.D. Thesis - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Image segmentation is a critical step in image analysis, and generally involves a high computational cost, especially when dealing with large volumes of data. Given the significant increase in the spatial, spectral and temporal resolutions of remote sensing imagery in the last years, current sequential and parallel solutions fail to deliver the expected performance and scalability. This work proposes a distributed image segmentation method, capable of handling very large high-resolution images in an efficient and scalable way. The proposed solution is based on the MapReduce model, which offers a highly scalable and reliable framework for storing and processing massive data in cluster environments and in private and public computing clouds. The proposed method is extendable to any region-growing algorithm and can be adapted to other models. The solution was implemented and validated using the Hadoop platform. Experimental results attest the viability of performing distributed segmentation over the MapReduce model through cloud computing.

Keywords

Remote Sensing; Image Analysis; Image Segmentation; Cloud Computing; Distributed Systems.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

A célebre expressão atribuída ao filósofo chinês Confúcio (551 a.C. - 479 a.C.) “Uma imagem vale mais que mil palavras” exprime a importância da visão computacional para o desenvolvimento humano. Uma imagem registra sinais emitidos pelo mundo real e contém uma infinidade de dados que, somente quando analisados corretamente, fornecem informações de extrema valia para a solução de inúmeros problemas em diferentes áreas como medicina, microscopia, segurança, robótica e sensoriamento remoto.

Numa descrição mais organizada, é possível definir visão computacional como a ciência responsável pela elaboração dos fundamentos teóricos e dos algoritmos necessários para extrair e analisar automaticamente informações sobre o mundo por intermédio de observações de uma ou mais imagens através de cálculos realizados por um computador (Haralick e Shapiro, 1992).

Nesse sentido, a análise de imagens é muitas vezes considerada como uma subárea da visão computacional cujo principal objetivo também está relacionado com a extração de dados ou informações de uma imagem por métodos automáticos ou semiautomáticos (Gonzalez e Woods, 2007). Pode-se dizer ainda que a análise de imagens geralmente se refere ao processamento computacional a fim de se gerar uma classificação dos objetos presentes na imagem (Pavlidis, 1988).

Entre as diversas áreas de aplicação, o sensoriamento remoto possui grande valor por possibilitar a avaliação e o monitoramento de regiões da Terra à distância. Para exemplificar a abrangência e importância desta área, cem diferentes usos e aplicações modernas do sensoriamento remoto podem ser encontrados em (GISGeography, 2015).

Praticamente restrito para fins militares no início, o sensoriamento remoto ganhou notável destaque em nível mundial após a alteração da política espacial americana com relação à geração e comercialização de dados de sensores orbitais ocorrida na década de 90, que passou a dar ênfase à promoção de plataformas

comerciais de sensoriamento remoto com resoluções espaciais submétricas. Além da mudança na estratégia de regulação, a nova política contemplou formas de incentivo à expansão do mercado que se antevia (Hitchings, 2003).

Esta nova postura do governo americano tomou forma tangível em 1999 com o lançamento do satélite IKONOS, primeiro satélite comercial a produzir imagens de altíssima resolução espacial (VHR – do inglês *Very High Resolution*). A partir de então, sensores orbitais com resoluções cada vez mais altas vêm sendo lançados.

Por consequência, a pesquisa em análise de imagens de altíssima resolução anteriormente fundamentada principalmente em imagens aéreas, caras e pouco acessíveis, recebeu um grande impulso a partir da última década. Em tempo, foi proposta uma mudança de paradigma relacionada com os métodos de análise de imagem capazes de explorar além de atributos espectrais, outros elementos interpretativos (p.ex., textura, forma ou contexto) aparentes em imagens de altíssima resolução. Este campo de pesquisa passou então a ser denominado de *Geographic Object Based Image Analysis*, ou simplesmente – GEOBIA (Blaschke, 2010; Blaschke et al., 2014, Hay e Castilla, 2008).

De uma maneira mais específica, é possível definir GEOBIA como uma subdisciplina da geoinformática focada no desenvolvimento de métodos automáticos para extração de atributos de objetos significativos a partir de imagens de sensoriamento remoto; na avaliação de suas características através do tempo e espaço; e na geração de novas geointeligências (i.e., conteúdo geoespacial sobre o contexto) a partir de múltiplos conjuntos de dados relacionados (Hay, 2014).

A primeira etapa das abordagens envolvendo GEOBIA geralmente é a segmentação de imagens em regiões homogêneas para uma posterior associação de rótulos na etapa de classificação. Desta forma, a segmentação é essencial no processo de análise de imagens baseada em objeto (Blaschke e Strobl, 2001). Diferentes algoritmos de segmentação têm sido propostos nas últimas quatro décadas (p.ex., Riseman e Arbib, 1977; Fu e Mui, 1981; Haralick e Shapiro, 1985; Pal e Pal, 1993; Deb, 2008). A avaliação da qualidade e a otimização dos parâmetros destes algoritmos também são objetos de pesquisa nesta área (Neubert et al., 2008; Happ et al., 2012).

O processo de segmentação de imagens geralmente envolve um custo computacional elevado (Wassenberg et al., 2009). 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 a principal opção para acelerar o procedimento de segmentação de imagens reduzindo consideravelmente o tempo gasto na análise de imagens baseadas em objetos (Moga et al., 2008, Montoya et al., 2003, Lenkiewicz et al. 2009, Happ et al., 2013).

No entanto, o crescimento na quantidade de imagens disponíveis, bem como o significante aumento nas resoluções espaciais, espectrais e temporais nos últimos anos tem exposto as limitações das técnicas atualmente aplicadas em análise de imagens resultando em uma nova demanda por soluções escaláveis (Vatsavai, 2013).

De fato, centenas de satélites se encontram em operação na órbita terrestre produzindo um grande volume de dados diariamente. Como exemplo recente, a Skybox Imaging colocou em funcionamento, no final de 2013, o SkySat-1 com capacidade de capturar vídeos em alta definição de até 90 segundos, a 30 quadros por segundo, sobre a superfície terrestre. A empresa, adquirida pela Google em 2014, já opera o SkySat-2 e pretende ter em um futuro próximo uma constelação de 24 satélites capaz de obter imagens de qualquer ponto da Terra até 3 vezes por dia (Space Flight Now, 2014). Cada um dos dois satélites atualmente em funcionamento produz cerca de 1 TB de dados por dia (Open Forum, 2015).

Em 2014, a Digital Globe foi autorizada a comercializar imagens com resoluções de até 40cm. Anteriormente a restrição era de 50cm. Este limite passou para 25cm em fevereiro de 2015 quando as imagens do WorldView-3, lançado em agosto de 2014 e com resolução de 30cm, passaram a estar disponíveis para venda (Reuters, 2014 e 2015). Em 2013, quando possuía cinco satélites, a Digital Globe coletava cerca de 2 PB por ano, além de manter 40 PBs em disco e 20 PBs em fitas (Networking Computing, 2013). Atualmente, apenas o WorldView-3 envia para Terra cerca de 1,2 GB por segundo (Apogeo Spatial, 2014).

O projeto EOSDIS (*Earth Observing System Data and Information System*) responsável pelos dados de observação da Terra da NASA gerenciou em 2014 mais de 9 petabytes. Além disto, estima-se que, diariamente, cerca de 6,4 TB de

dados são adicionados a seus arquivos e quase 28 TB são distribuídos para uma média de 11 mil usuários únicos em todo mundo (Earthdata, 2015).

Quando se trata de gerir grandes volumes de dados, a computação em nuvem (*cloud computing*) tem se destacado tanto no cenário acadêmico quanto no industrial e no governamental provando ser uma plataforma poderosa para realizar computações complexas e de larga escala (Hashem et al., 2015). Esta plataforma tem como principal vantagem o fornecimento de recursos computacionais sob demanda provendo uma escalabilidade praticamente infinita a custos reduzidos.

Desta forma, a computação em nuvem oferece um ambiente com capacidade de armazenar e processar uma enorme quantidade de dados sem a necessidade de se adquirir e manter uma infraestrutura computacional sofisticada, o que antes só era possível para grandes organizações. O usuário passa a focar nos seus próprios objetivos sem a necessidade de se preocupar com problemas relacionados a infraestrutura, flexibilidade e disponibilidade de recursos (Aceto et al., 2013).

Recentemente, a Amazon anunciou a disponibilidade de mais de 85.000 cenas do Landsat 8 em seu serviço de armazenamento na nuvem. Todas as cenas de 2015, junto com uma seleção de cenas de 2013 e 2014 podem ser acessadas sem custos. Novas imagens são disponibilizadas a cada dia geralmente após horas de produção. Esta é uma iniciativa em conjunto com a United States Geological Survey (USGS) que gerencia o Programa Landsat e torna os dados disponíveis gratuitamente (Amazon Web Services, 2015-a).

Em relação ao processamento, o MapReduce (Dean e Ghemawa, 2004) é um dos modelos de programação mais utilizados para processar grandes volumes de dados na nuvem e tem sido aplicado em diversas áreas (Assunção, 2014). O modelo fornece uma estrutura eficiente, altamente escalável e tolerante a falhas ocultando detalhes da execução paralela e permitindo que os usuários se concentrem apenas nas estratégias de processamento de dados. Vale ainda mencionar o Hadoop (Apache Hadoop, 2015), que por ser de código aberto, é a implementação mais popular do MapReduce. Uma variedade de instituições que vem utilizando esta plataforma pode ser encontrada em (Hadoop Wiki, 2015).

Apesar de ser bastante utilizado, o Hadoop pode ser considerado um projeto relativamente recente. Diversas ferramentas vêm sendo desenvolvidas a fim de otimizar e facilitar sua utilização, como é o caso do Pig (Apache Pig, 2015) que

fornece, entre outros benefícios, maior facilidade de programação e extensibilidade através de funções definidas pelos usuários (UDFs).

No início de 2015, a Google anunciou o MapReduce for C (MR4C): uma plataforma de código aberto que permite a execução de código nativo em C/C++ no Hadoop. A iniciativa foi originalmente desenvolvida pela Skybox Imaging e tem justamente como motivação facilitar o processamento em larga escala de imagens de satélite e de dados geoespaciais (Google Open Source Blog, 2015).

Este fato deve facilitar e incentivar o desenvolvimento de soluções de sensoriamento remoto usando o Hadoop, o que ainda é pouco explorado nesta área. Quando se trata de GEOBIA então, raros são os trabalhos encontrados.

Esta tese está inserida neste contexto com o intuito de aproveitar os benefícios advindos da computação em nuvem para realizar o processamento distribuído da segmentação de imagens. Assim, este trabalho propõe um método distribuído de segmentação de imagens baseado em MapReduce.

A solução apresentada é eficiente, escalável e de baixo custo; e se destaca por possibilitar o processamento de imagens grandes de altíssima resolução, o que antes não era possível. Além disto, ela pode ser aplicada para qualquer algoritmo de crescimento de regiões e também ser adaptada para outras técnicas de segmentação.

1.1. Objetivos

O objetivo principal desta tese é propor e desenvolver um método distribuído de segmentação de imagens capaz de tratar grandes imagens de altíssima resolução. Este método deve ser adequado para computação em nuvem a fim de fornecer uma solução escalável, robusta e computacionalmente eficiente.

São objetivos específicos deste trabalho:

- Propor um método de segmentação distribuído, baseado em crescimento de regiões, que seja capaz de produzir resultados sem a presença de artefatos.
- Implementar o método proposto, criando uma ferramenta distribuída que possa ser executada em um ambiente de computação em nuvem.

- Avaliar a escalabilidade, o desempenho computacional e a qualidade dos resultados alcançados pelo método proposto dado um conjunto de imagens de sensoriamento remoto.
- Fornecer meios para integração do método proposto com uma plataforma distribuída de análise de imagens na nuvem, como o InterIMAGE Cloud Platform (Ferreira, 2014).

1.2.Contribuições originais da tese

Além de apresentar um novo método distribuído para segmentação de imagens que possibilite o processamento de grandes imagens de sensoriamento remoto na nuvem, esta tese apresenta as seguintes contribuições:

- Uma avaliação do Hadoop/Pig como plataforma para o desenvolvimento de operadores de processamento de imagem na nuvem.
- Um novo método para a segmentação distribuída de imagens para execução em *clusters* de computadores e em nuvens privadas ou comerciais, baseada no modelo MapReduce, capaz de processar um grande volume de dados.
- Um método de segmentação distribuída que não introduz artefatos decorrentes do processamento distribuído.
- Uma ferramenta em *software* livre para a segmentação distribuída de imagens, resultante da aplicação da metodologia proposta à versão sequencial de um algoritmo amplamente utilizado pela comunidade de sensoriamento remoto.
- A integração da ferramenta desenvolvida ao InterIMAGE Cloud Platform, com o objetivo de suprir uma necessidade real do sistema, que por si só, também é uma plataforma original para execução automática de modelos de interpretação de imagens na nuvem.

1.3. Estrutura da tese

O restante desta tese se encontra dividido em seis capítulos organizados da seguinte forma:

- Capítulo 2 – Neste capítulo, são expostos fundamentos teóricos importantes para a compreensão da tese e dos trabalhos relacionados. São descritas diferentes técnicas de segmentação de imagens passíveis de serem utilizadas. Também é apresentada uma breve explanação sobre computação paralela, além dos conceitos fundamentais de MapReduce, Hadoop e Pig.
- Capítulo 3 – Este capítulo apresenta uma revisão da literatura descrevendo trabalhos relacionados aos objetivos da tese.
- Capítulo 4 – O capítulo 4 descreve o método de segmentação distribuída proposto no presente trabalho e apresenta algumas de suas características.
- Capítulo 5 – Este capítulo expõe uma avaliação experimental relacionada com o desempenho computacional e com a qualidade dos resultados gerados pelo método proposto.
- Capítulo 6 – As principais conclusões deste trabalho, bem como indicações de trabalhos futuros são detalhadas neste capítulo.

2
Fundamentos teóricos

Este capítulo aborda os fundamentos teóricos importantes para o entendimento do método proposto nesta tese e exposto no capítulo 4. Algumas definições auxiliam também no entendimento dos trabalhos relacionados listados no capítulo 3. A seguir, são apresentados conceitos sobre segmentação de imagens, computação paralela e as estruturas de MapReduce, Hadoop e Pig.

2.1. Segmentação de imagens

A segmentação de imagens tem como objetivo a identificação de regiões que representem objetos de interesse (Russ, 1998). Desta forma, este processo consiste na subdivisão da imagem em regiões cujos *pixels* compartilham alguma propriedade (Gonzalez e Woods, 2007).

Esta é uma etapa essencial para o processo de análises de imagens uma vez que ela é responsável por delinear as regiões que irão ser posteriormente classificadas por especialistas ou por algoritmos de classificação. Desta forma, a qualidade da segmentação tem influência direta no resultado final do processo de análise de imagens. No entanto, não existe uma teoria geral sobre o assunto e diferentes técnicas são sugeridas de acordo com o domínio do problema.

Na verdade, a segmentação de imagens é tida como um problema *ill-posed*. De acordo com (Hadamard, 1902), para um problema ser *well-posed* sua solução deve atender a três critérios: existir, ser única e ser robusta a pequenas mudanças nas condições iniciais. No caso da segmentação, dependendo do algoritmo envolvido, a inclusão de poucas linhas ou colunas na imagem, ou ainda a mudança da localização da origem, através do espelhamento da imagem por exemplo, podem alterar o resultado. Até mesmo quando foto-intérpretes humanos são considerados, os resultados dificilmente são idênticos.

De maneira geral, as técnicas de segmentação se baseiam na detecção de descontinuidades ou similaridades. A primeira é relacionada com a existência de uma varição abrupta de intensidade nas proximidades da fronteira dos segmentos.

Já a segunda, pressupõe que os *pixels* que fazem parte de um mesmo segmento são semelhantes de acordo com algum critério definido. Entre as técnicas existentes, podemos citar: a segmentação por detecção de bordas, por limiares, por divisores de água, por clusterização e por crescimento de regiões. Estas técnicas são brevemente descritas a seguir.

2.1.1. Segmentação por detecção de bordas

A segmentação baseada na detecção de bordas busca identificar alterações bruscas na intensidade dos *pixels*. Essas alterações geralmente evidenciam a presença de bordas, chamadas de arestas, que quando conectadas formam as fronteiras dos segmentos.

Geralmente as descontinuidades são detectadas por intermédio de derivadas de primeira e/ou de segunda ordem. No caso da primeira derivada, uma elevada magnitude do gradiente evidencia uma aresta enquanto na segunda derivada esta é localizada no cruzamento em zero.

O cálculo das derivadas pode ser realizado através de filtros lineares utilizando máscaras para a varredura da imagem. Operadores de Sobel (Sobel, 1990), Prewitt (Prewitt, 1970) e laplacianos são usados para este fim. A aplicação do algoritmo de Canny (Canny, 1986) também é comum para descartar *pixels* irrelevantes (assiciados a bordas fracas) e para conectar segmentos de bordas identificados pelos filtros mencionados.

2.1.2.Segmentação por limiares

A segmentação por limiares, ou limiarização, é uma das técnicas mais simples de segmentação e classifica os *pixels* de uma imagem de acordo com sua intensidade obedecendo a um ou mais limiares. A segmentação é realizada a partir do histograma da imagem, onde os *pixels* são agrupados conforme sua intensidade, obedecendo ao limiar ou limiares propostos.

Em sensoriamento remoto, é comum a utilização de índices normalizados para a detecção de algumas classes específicas como vegetação (NDVI - *normalized difference vegetation índex*), água (NDWI - *normalized difference water índex*) e solo exposto (NDBSI - *normalized difference bare soil índex*) por meio da limiarização. Esses índices são baseados em combinações aritméticas de

duas ou mais bandas espectrais da imagem. Os limiares são geralmente definidos de acordo com o conhecimento de especialistas considerando aplicações específicas.

2.1.3.Segmentação por divisores de água (*Watershed*)

Nesta técnica, a imagem é vista como uma superfície topográfica composta de vales e montanhas onde o valor da magnitude do gradiente relativo à intensidade dos *pixels* corresponde à altitude dos pontos (Beucher e Meyer, 1993).

Um processo simula a inundação dessa superfície a partir dos mínimos locais, formando bacias ou poças. Quando as águas de duas bacias vizinhas estão na iminência de fazerem contato para transformarem-se em uma única bacia, uma linha de contenção é criada, formando os contornos e por fim as bordas de cada segmento resultante (Pedrini e Schwartz, 2008).

Assim, pode-se dizer que os segmentos são formados por regiões que, partindo de um mínimo local, formam uma bacia hidrográfica.

2.1.4.Segmentação por clusterização

A segmentação por clusterização é baseada na aglomeração de um conjunto de pixels em clusters de acordo com as propriedades espectrais dos pixels. O algoritmo *K-means* (MacQueen, 1967) é um exemplo de aplicação desta técnica. Neste caso, um número de grupos é pré-definido e, de forma iterativa, o algoritmo identifica os centros de massa de cada grupo e os atualiza.

Outro algoritmo bastante utilizado é o de deslocamento da média (Mean-Shift). O *Mean-Shift* pode ser definido como um estimador não paramétrico de gradiente que não requer o conhecimento prévio do número de agrupamentos e não restringe a forma dos clusters (Comaniciu e Meer, 2002). A aplicação desta técnica para segmentação de imagens pode ser feita de forma quase direta, representando cada pixel como um vetor de atributos.

2.1.5.Segmentação por regiões

Os métodos de segmentação por regiões visam agrupar *pixels* adjacentes, segundo algum critério de similaridade, no intuito de gerar ao final do processo

regiões internamente homogêneas. As regiões podem ser agrupadas ou divididas com base em suas propriedades espectrais, morfológicas ou texturais. Dentre os métodos desta categoria destacam-se as técnicas de crescimento de regiões (*region growing*).

Crescimento de regiões (*region growing*)

Neste método, um conjunto de *pixels* chamados sementes é escolhido inicialmente. A partir das sementes, as regiões crescem através da agregação de outros *pixels* ou regiões vizinhas, atendendo a determinado critério de homogeneidade. As sementes podem ser selecionadas de maneira aleatória, determinística, ou pelo usuário (Pedrini e Schwartz, 2008).

Algumas dificuldades encontradas neste método são a definição do critério de parada para o crescimento das regiões, a dependência dos resultados em relação à escolha das sementes e a determinação das características apropriadas para utilização como critério de homogeneidade.

2.2.Algoritmo de segmentação utilizado

Dentro do grande número de algoritmos de segmentação de imagens existentes, um algoritmo de crescimento de regiões foi selecionado para validar o funcionamento do método desenvolvido nesta tese: o algoritmo proposto em (Baatz e Schäpe, 2000). Este algoritmo é um dos mais usados nos anos recentes na área de sensoriamento remoto (Neubert et al, 2008).

Além disso, ele envolve grande dependência de dados, o que requer uma maior complexidade no desenvolvimento de estratégia distribuída. Assim, se o método proposto suprir a necessidade deste algoritmo, também atenderá às necessidades dos algoritmos mais simples.

Este algoritmo 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.

A fim de 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 2.1, é definido pela soma ponderada de um componente referente à heterogeneidade espectral (h_{cor}) e outro relacionado à heterogeneidade morfológica (h_{forma}). O cálculo de ambos (h_x) se baseia na diferença entre o possível objeto gerado (*obj3*) e a soma independente dos objetos geradores (*obj1* e *obj2*) como mostra a Equação 2.2. É possível ainda definir a importância relativa entre estes componentes (w_{cor}).

$$f = w_{cor}.h_{cor} + (1 - w_{cor}).h_{forma} \tag{2.1}$$

$$h_x = h_{obj\,3} - (h_{obj\,1} + h_{obj\,2}) \tag{2.2}$$

A heterogeneidade espectral (h_{cor}) é dada pela soma ponderada do desvio padrão da intensidade de cada *pixel* do segmento. A contribuição de cada uma das bandas espectrais na formação da heterogeneidade espectral também pode ser ponderada.

A heterogeneidade morfológica (h_{forma}), por sua vez, é composta por dois elementos diferentes: compacidade e suavidade. A compacidade (*Cmp*), formulada na Equação 2.3, 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 2.4. Também é possível definir a importância relativa entre a compacidade e a suavidade na composição da heterogeneidade morfológica.

$$Cmp = \frac{b}{\sqrt{n}} \tag{2.3}$$

$$Svd = \frac{b}{\sqrt{bbox}} \tag{2.4}$$

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.

Vale ainda mencionar que, após realizar o cálculo do custo de fusão entre um objeto e seus vizinhos, é necessária uma heurística que defina qual dos segmentos vizinhos deve ser escolhido para fusão. Quatro alternativas são apresentadas e descritas a seguir:

- O procedimento mais simples, denominado neste trabalho de "vizinho aleatório" (*fitting*), admite a fusão de um segmento com qualquer um de seus segmentos vizinhos que atenda ao critério de heterogeneidade.
- O método do "melhor vizinho" (*best fitting*) estabelece que um segmento deve ser fundido com o segmento vizinho que, além de atender ao critério de heterogeneidade, resulta no menor aumento de heterogeneidade entre todos os vizinhos.
- A decisão pelo "melhor vizinho mútuo" (l*ocal mutual best fitting*) é atendida quando 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.
- A última heurística, denominada "melhor vizinho mútuo global" (g*lobal mutual best fitting*), determina que somente o par de melhores vizinhos mútuos que resulte no menor aumento de heterogeneidade poderá ser fundido.

Vale lembrar que, em todas as alternativas mencionadas, a fusão só se consuma se o aumento de heterogeneidade decorrente da fusão não exceder o valor atribuído ao parâmetro de escala (s).

2.3. Computação paralela

A necessidade de lidar com problemas cada vez mais complexos gerou uma crescente demanda por desempenho na comunidade técnico-científica. São necessários computadores cada vez mais potentes de forma a viabilizar certas aplicações computacionalmente custosas.

A tecnologia dos processadores evoluiu bastante nas últimas décadas provendo um aumento considerável de desempenho. Contudo, restrições físicas como a dissipação de energia e calor fizeram com que estes avanços atingissem um limite (Asanovic et al., 2009). Assim, apesar do conceito de computação paralela não ser novidade, a exploração de arquiteturas e algoritmos voltados ao processamento paralelo foi impulsionada na tentativa de contornar esta barreira.

O processamento paralelo consiste na utilização simultânea de diferentes recursos computacionais para a execução de trechos independentes de uma determinada tarefa. Atualmente, o grande progresso tecnológico neste setor resultou em uma redução de custos de arquiteturas específicas que comportam o paralelismo.

2.3.1.Multiprocessadores simétricos

Um único computador pode apresentar dois ou mais processadores similares de capacidade comparável que compartilham o mesmo espaço de endereçamento de memória e dispositivos de entrada e saída. É possível ainda que mais de um processador esteja encapsulado em um mesmo chip de forma a resultar nos chamados múltiplos núcleos (*multicore*).

O método mais simples de explorar o paralelismo desta arquitetura é através da execução simultânea de diferentes *threads* nos diferentes processadores. Quando se trata do paralelismo de um único processo, é necessário dividir o processamento entre os múltiplos processadores e núcleos através de diferentes linhas de execução ou *threads*. Cada execução é independente e, devido ao compartilhamento de memória, a sincronização e controle de execução entre elas são facilitados.

Nesse tipo de arquitetura é possível adicionar processadores a fim de aumentar o desempenho computacional, contudo essa é uma solução não escalável tendo em vista que construir uma memória compartilhada para um grande número

de processadores é inviável, além de causar grandes problemas de contenção de memória.

2.3.2.Unidades de processamento gráfico (GPUs)

O progresso das unidades de processamento gráfico (GPUs – *Graphic Processing Units*) foi motivado pelo mercado de jogos que necessitava de equipamentos capazes de realizar cálculos e tarefas relacionadas ao processamento de gráficos em tempo real.

Rapidamente percebeu-se o potencial desta arquitetura para computação paralela de forma a estender o seu uso para diversas finalidades que buscavam alto desempenho. Surgiu assim a Computação de Propósito Geral em GPUs (GPGPU), que se refere ao aproveitamento da capacidade de processamento das unidades de processamento gráfico para aplicações de propósito geral, ou seja, na utilização de GPUs para aceleração de tarefas não gráficas.

De uma forma geral, uma GPU pode ser vista como um processador de diversos núcleos cuja capacidade computacional provém de uma arquitetura altamente paralela. Assim, a GPU trabalha com divisão por *threads* em diferentes núcleos localizados em elementos processadores. As *threads* também compartilham um espaço de memória em blocos de execução e um espaço de memória global.

Geralmente existem centenas de elementos processadores a fim de prover o alto paralelismo. Porém, esses processadores apresentam uma arquitetura relativamente simples, o que dificulta a programação e limita os tipos de algoritmos que podem se beneficiar da utilização desta arquitetura. Por consequência, o paralelismo deve ter uma granularidade fina a fim de se obter um melhor desempenho computacional. Mais informações sobre computação em GPUs podem ser encontradas em (Owens et al., 2008)

2.3.3.Multicomputadores

Neste tipo de arquitetura o processamento é distribuído entre dois ou mais computadores. Cada computador possui seu próprio espaço de endereçamento de memória e a comunicação entre eles deve ser realizada através de algum tipo de troca de mensagens. Este fato torna a tarefa de programação mais complexa, no

entanto, confere uma solução relativamente escalável, tendo em vista que é possível a inclusão de novos computadores.

Geralmente os computadores são organizados em *clusters*. Um *cluster* pode ser definido como um grupo de computadores interligados trabalhando em conjunto como um recurso unificado, criando a ilusão de ser uma única máquina (Stallings, 2009).

2.4.Computação em nuvem (*cloud computing*)

Computação em nuvem, ou *cloud computing*, geralmente se refere ao fornecimento de recursos de *software* e/ou *hardware* através da internet, por meio de serviços, de acordo com a necessidade.

Neste sentido, em (Breitman e Viterbo, 2010) há uma comparação deste modelo com a evolução da produção e fornecimento de energia. Se durante o período da Revolução Industrial as grandes fábricas eram responsáveis por gerar sua própria energia elétrica e/ou mecânica, atualmente elas adquirem energia como um serviço cobrado pela quantidade consumida. A computação em nuvem tende a um progresso semelhante onde os recursos computacionais serão de responsabilidade de algumas empresas especializadas que fornecerão estes serviços sob demanda.

Apesar de algumas divergências em relação ao surgimento do termo *cloud* para expressar este modelo, foi em 2006 que ele ganhou popularidade. Neste ano, tanto a Google quanto a Amazon utilizaram este termo para descrever o funcionamento de alguns de seus serviços. Ainda em 2006, um artigo publicado na Wired ganhou destaque (Gilder, 2006) ao "dar as boas-vindas à Internet Cloud no alvorecer da era do petabyte".

As definições para computação em nuvem divergem de certa forma. Após pesquisar e avaliar diversas proposições, (Vaquero et al., 2009) propõe a seguinte: um grande conjunto de recursos virtuais facilmente usáveis e acessíveis, tais como *hardware*, plataformas de desenvolvimento e serviços. Estes recursos podem ser dinamicamente reconfigurados para se ajustarem a uma carga variável e são geralmente explorados por meio de um modelo de pagamento de acordo com o uso.

De acordo com o NIST (National Institute of Standards and Technology) (Mell e Grance, 2011), a computação em nuvem é um modelo que possibilita o acesso à rede, sob demanda, a um conjunto compartilhado de recursos computacionais configuráveis (e.g., servidores, armazenamento, rede, aplicações, serviços) que podem ser rapidamente provisionados e entregues com o mínimo esforço de gerenciamento ou de interação do provedor de serviços.

A computação em nuvem geralmente é classificada de acordo com a implantação e o tipo de serviço fornecido. Em relação à implantação, as nuvens são normalmente descritas como públicas, privadas ou híbridas. O modelo a ser utilizado deve ser escolhido de acordo com a necessidade das aplicações. Enquanto uma nuvem pública é compartilhada entre diversos usuários, a privada é de uso exclusivo, geralmente uma empresa. A nuvem híbrida é uma combinação dos dois casos (Dikaiakos et al. 2009).

Em relação aos serviços, três classes de modelos são propostas: *Infraestructure as a Service* (IaaS), *Plataform as a Service* (Paas) e *Software as a Service* (SaaS). O primeiro modelo fornece a possibilidade de utilização de computadores (físicos ou virtuais) e outros recursos com a possibilidade de inclusão dinâmica de novos servidores virtuais à infraestrutura utilizada. Um exemplo é o Amazon Elastic Compute Cloud (EC2).

O modelo PaaS oferece plataformas computacionais que fornecem ambientes de execução de linguagens de programação, banco de dados, servidores web entre outros. Neste caso, não há a preocupação de gerenciar e escalar os recursos manualmente, tornando a camada de infraestrutura transparente. Um exemplo deste modelo é Google AppEngine.

Por fim, o modelo SaaS fornece o *software* ao usuário juntamente com a infraestrutura e a plataforma necessária para sua execução, que são transparentes. Um exemplo de aplicação deste modelo é o GoogleApps. Em todos os casos, o sistema de cobrança dos serviços geralmente está relacionado com a quantidade de recursos alocados sob demanda pelo usuário.

O modelo de computação na nuvem também compreende cinco características essenciais segundo o NIST (Mell e Grance, 2011):

- *Self-service sob demanda*: o cliente pode alocar serviços automaticamente, de acordo com a necessidade, sem que seja preciso qualquer interação humana com o provedor.

- *Amplo acesso via rede*: recursos estão disponíveis em rede e podem ser acessados de qualquer lugar por diferentes plataformas (celulares, *tablets*, *notebooks*, estações de trabalho, etc.)
- *Agrupamento de recursos*: o provedor de serviços deve alocar dinamicamente recursos físicos e virtuais de acordo com a demanda a fim de atender múltiplos clientes.
- *Elasticidade rápida*: O fornecimento de recursos deve ser elástico de forma a aumentar e reduzir rapidamente a quantidade aprovisionada de acordo com a demanda, criando a ilusão de recursos ilimitados e disponíveis a qualquer momento.
- *Medição de serviço*: O uso de recursos deve ser monitorado, controlado e apresentado de forma a proporcionar transparência, tanto para o provedor quanto para o consumidor dos serviços

Assim, uma das principais vantagens da computação em nuvem é a capacidade de acesso aos dados de qualquer local através da Internet. Além disto, o custo é reduzido e a cobrança é realizada somente sobre o que for consumido, sem gastos para aquisição e manutenção de uma infraestrutura física. Adicionalmente, é possível aumentar ou diminuir a quantidade de recursos de acordo com a necessidade.

O grande desafio existente diz respeito à privacidade e segurança dos dados, principalmente quando se utiliza nuvens públicas (Sosinsky, 2011). Outras dificuldades existentes estão relacionadas ao desenvolvimento de aplicações que sejam escaláveis e que os serviços disponíveis sejam confiáveis de forma a serem tolerantes a falhas e estarem sempre disponíveis. Por fim, o custo computacional derivado da comunicação realizada entre os recursos através da rede é outro fator a ser considerado.

Dentro da computação em nuvem, o modelo MapReduce e a plataforma Hadoop se destacam quando se trata de processar grandes volumes de dados. Estes conceitos são descritos a seguir.

2.5. MapReduce e Apache Hadoop

Hadoop é uma plataforma de software para o desenvolvimento de aplicações que processam grandes quantidades de dados em paralelo em *clusters* de

computadores de uma maneira confiável e tolerante a falhas (Apache Hadoop, 2015).

Processar grandes volumes de dados de forma distribuída não é novidade, porém o modelo de programação simplificada do Hadoop permite ao usuário escrever e testar rapidamente sistemas distribuídos. Além disso, a distribuição de trabalho e de dados entre as máquinas é realizada de forma automática e eficiente podendo inclusive aproveitar o paralelismo existente nos núcleos das CPUs (Yahoo!, 2015).

O Hadoop se baseia em dois conceitos principais: o sistema de arquivos distribuídos (HDFS - *Hadoop Distributed File System*) e o modelo de programação MapReduce. O HDFS é projetado e otimizado para alto rendimento e funciona melhor quando trabalha com grandes arquivos (gigabytes ou maiores) (Holmes, 2012).

O intuito do HDFS é dividir grandes arquivos de dados em pedaços ou *chunks* que são controlados por diferentes nós do cluster. Assim, cada processo em execução em um nó no cluster executa um subconjunto desses registros. A plataforma Hadoop escalona os processos de acordo com a localização dos dados adotando uma estratégia de mover o processamento para a localidade do dado a fim de poupar transferências desnecessárias e alcançar um maior desempenho. Esta estratégia é possível devido ao conhecimento prévio existente no HDFS. Além disso, cada pedaço pode ser replicado em várias máquinas, de modo que uma falha não resulte em quaisquer indisponibilidades de dados.

O Hadoop Mapreduce está relacionado com o modelo de programação. MapReduce é um paradigma de processamento de dados simples, porém poderoso que foi proposto e utilizado pelo Google (Dean e Ghemawa, 2004). Geralmente os *jobs* MapReduce são compostos por três fases principais: *map*, *shuffle* e *reduce* (Gates, 2011).

Na fase de *map,* o processamento é realizado em cada registro da entrada separadamente, definido por um par chave-valor. Na fase de *shuffle*, os dados são aglomerados de acordo com a chave utilizada e então distribuídos para máquinas diferentes a fim de iniciar a fase de *reduce*. Registros que contenham a mesma chave são enviados para o mesmo processo *reduce*, tornando possível a execução combinada dos diferentes resultados a fim de gerar a saída. Este processamento é realizado em paralelo em diferentes nós do cluster (Figura 2.1).

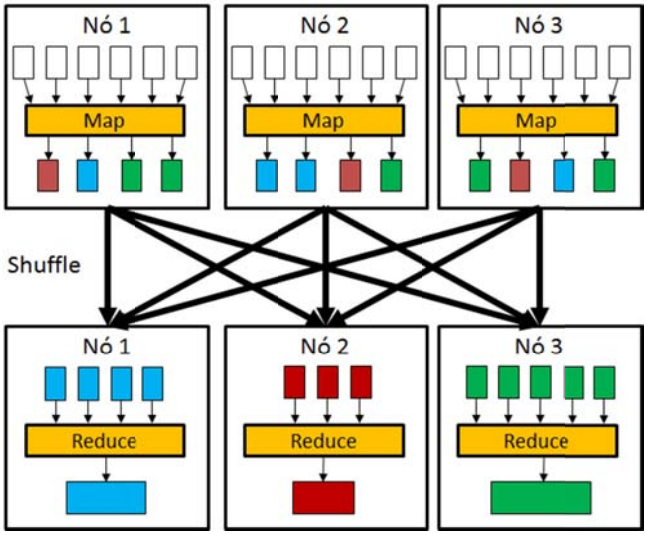


Figura 2.1. Etapas de Map, Shuffle e Reduce em diferentes nós do cluster.

A execução dos *Jobs* é controlada por dois tipos de nós: um *JobTracker* e diversos *TaskTrackers*. O *JobTracker* é o responsável por escalonar as tarefas a serem executadas nos *TaskTrackers*, controlar quais tarefas estão sendo executadas, quais são os recursos utilizados, etc. Os *TaskTrackers* se encarregam então da execução das tarefas de *map* ou de *reduce* e reportam todo o progresso ao *JobTracker* (White, 2012).

Apesar de possuir diversas facilidades, a programação em MapReduce pode apresentar complicações quando se deseja tratar de problemas mais complexos. Assim, diferentes alternativas foram apresentadas de forma a tornar esta tarefa mais simples e compreensiva aos programadores por intermédio de linguagens de alto nível como Pig, Hive, Cascading, Cascalog ou Crunch. Nesta tese, o interesse recai sobre Pig, que será brevemente descrito a seguir.

2.6.Apache Pig

O Pig fornece um motor para a execução distribuída de fluxos de dados sobre o Hadoop, além de uma linguagem, conhecida como Pig Latin, para expressar esses fluxos de dados. A linguagem Pig Latin inclui operadores para muitas das operações de dados tradicionais (*join*, *sort*, *filter*, etc), bem como a capacidade para usuários desenvolverem suas próprias funções para leitura, processamento e escrita de dados (Gates, 2011).

Desta forma, Pig apresenta uma camada para compilação em MapReduce e uma camada de linguagem. Isto fornece maior facilidade de programação, possibilidade de otimização na criação de *jobs* MapReduce e extensibilidade através das UDFs (*User Defined Functions*), que são funções definidas pelos usuários.

Atualmente as UDFs estão disponíveis em seis linguagens: Java, Jython, Python, JavaScript, Ruby e Groovy, portanto, funções desenvolvidas nestas linguagens podem ser facilmente integradas ao Pig. Isto fornece uma forma simples e eficaz para a implantação de novas funcionalidades a serem executadas em paralelo, sem a necessidade de lidar com a complexidade da linguagem de programação MapReduce (Apache Pig, 2015).

Além disso, durante a compilação do script Pig Latin, a plataforma Pig analisa todo o fluxo de processamento a fim de otimizar os *jobs* a serem criados.

3
Trabalhos relacionados

Dada a importância da segmentação de imagens, inúmeros trabalhos relacionados ao tema foram publicados ao longo das últimas décadas. É possível, por exemplo, encontrar na literatura uma grande quantidade de técnicas de segmentação de imagens (Skarbek e Koschan, 1994; Dey et al., 2010; Vantaram e Saber, 2012; Narkhede, 2013).

Por conta desta diversidade, diferentes estudos envolvendo a comparação entre as técnicas disponíveis foram realizados (Neubert e Meinel, 2003; Byrd et al., 2005; Neubert et al., 2008; Heimann et al., 2009, Marpu et al.,2010). Como consequência, diversas métricas de avaliação de qualidade também foram propostas (Zhang, 1996; Correia e Pereira, 2000; Cardoso e Corte-Real, 2005; Zhang et al., 2008, Pont-Tuset e Marques, 2013). A escolha por uma dentre as técnicas existentes sempre está relacionada com o escopo de uma dada aplicação.

Também depende da aplicação a determinação dos diferentes parâmetros existentes em cada algoritmo. Assim, outro tema correlato é a análise destes parâmetros e a otimização dos mesmos através de métodos automáticos (Pignalberi et al., 2003; Feitosa et al., 2006; Happ et al., 2012, Achanccaray et al., 2014).

Em se tratando da aplicação de algoritmos de segmentação, existem igualmente uma infinidade de trabalhos. Em trabalhos recentes é possível encontrar aplicações na detecção de câncer (Halder et al., 2014), de tumores (Yan et al, 2014), na classificação de cobertura urbana (Al-Hameedawi e Buchroithner, 2014), detecção de culturas agrícolas (Xin Cao et al., 2014), na avaliação ambiental (Mei-ping Song et al., 2014) entre tantas outras.

3.1.Abordagens paralelas para segmentação de imagens

O custo computacional elevado de certos métodos de segmentação de imagens motiva a busca por soluções que explorem técnicas de paralelismo a fim

de obter maior desempenho computacional. Algumas destas propostas são apresentadas a seguir.

Uma antiga abordagem envolvendo crescimento de regiões em um supercomputador pode ser vista em (Singh et al., 1999). Inicialmente, um conjunto de sementes é pré-definido e distribuído para cada processador. O algoritmo de crescimento de regiões, com algumas modificações a fim de reduzir a dependência de dados, é executado por cada processador realizando as devidas comunicações quando necessário. A divisão da imagem em diferentes partições pode reduzir a quantidade de comunicação envolvida no tratamento das sobreposições de segmentos. Um mecanismo de controle de crescimento é utilizado para que uma região que seja processada mais rapidamente não cresça de forma anormal.

Uma proposta para segmentação por divisão e união é descrita em (Montoya et al., 2003) e consiste na divisão da imagem em partições a serem processadas em diferentes processadores. Inicialmente, realiza-se a etapa de divisão baseada em quadrantes, que é inerentemente paralela. Cada processador é responsável por uma quantidade de quadrantes adjacentes. Em seguida, é gerado um grafo envolvendo o conjunto de arestas das regiões que satisfaçam os critérios de proximidade e homogeneidade em cada processador. Por intermédio da comunicação entre os processadores, o grafo é então atualizado levando em consideração as fronteiras das diferentes partições. A partir deste ponto tem início a união das regiões considerando os melhores canditados locais e sua respectiva validação por intermédio da comunicação entre os processadores.

Dois algoritmos paralelos baseados em divisor de águas são apresentados em (Moga et al., 1998). Salvo algumas diferenças, a imagem é dividida em partições que contêm áreas de sobreposição entre si e são processadas em diferentes processadores. Em cada partição, um caminho é percorrido para cada *pixel* não marcado até a região de inserção. Apenas os *pixels* que satisfazem a relação de ordem definida no algoritmo são incorporados na região. Seu rótulo é então associado e propagado de forma reversa através do caminho. No final, os resultados são mesclados para eliminar as sobreposições e definir rótulos para a imagem como um todo.

A utilização de unidades de processamento gráfico (GPUs) para segmentação de imagens também se faz presente na literatura. No entanto, a

maioria das soluções disponíveis é focada para o tratamento de imagens médicas, que apresentam técnicas diferentes devido a suas distintas características e em relação às imagens de sensoriamento remoto. Este é o caso do proposto em (Baggio, 2007): um algoritmo paralelo interativo para a técnica *Livewire*. Também focado em imagens médicas, um algoritmo de segmentação volumétrico de imagens 3D para GPU, é sugerido em (Schenke et al., 2005), assim como em (Pan et al., 2008). Uma proposta paralela para a técnica *K-means* é apresentada em (Backer et al., 2013). O algoritmo é composto por duas etapas completamente paralelas relativas à clusterização e agregação dos componentes conectados existentes,

Uma abordagem para *clusters* de computadores é proposta em (Palomera-Perez et al., 2010). O primeiro passo envolve a definição de algumas sementes distribuídas ao longo da imagem. Em seguida, a imagem é dividida em grupos de recortes horizontais e verticais que são executados no *cluster*. Por existir uma área de sobreposição entre os recortes, um operador binário é utilizado para combinar a segmentação horizontal com a vertical e gerar um único resultado.

3.2.Segmentação de imagens envolvendo grande volume de dados

Apesar da aceleração de desempenho obtida nas abordagens descritas na seção anterior, existe um limite relacionado com o crescente aumento do tamanho das imagens e por consequência do volume de dados a ser processado. Assim, o grande desafio atual é a definição de soluções que possam processar de forma eficiente, sem problemas relacionados com a capacidade de memória do sistema.

Assim, preocupado com o crescente tamanho das imagens, em (Michel et al., 2012) propõe-se uma solução para a segmentação *Mean-Shift* ao dividir a imagem em recortes que sempre cabem na memória e processá-los de forma completamente independente através de múltiplas *threads*. Uma etapa posterior tenta refinar os resultados através da fusão de segmentos vizinhos de recortes adjacentes cuja área de contato é superior a um determinado valor. Com isto, é possível processar imagens grandes, porém o tempo de processamento cresce consideravelmente de acordo com o tamanho da imagem. Experimentos foram realizados com uma imagem do satélite Pléiades de 40.000 x 40.000 *pixels* e o resultado apresenta diversos artefatos nas bordas dos recortes.

Geralmente quando se trata de processar grandes volumes de dados, arquiteturas modernas escaláveis se sobressaem por serem as únicas que conseguem lidar com tamanha quantidade. Conforme descrito anteriormente, o modelo MapReduce tem sido bastante utilizado nas mais diversas áreas para a computação de larga escala. No entanto, poucos são os trabalhos relacionados com segmentação de imagens que foram desenvolvidos até o momento aderindo a esta nova plataforma.

Tesfamariam (2011) apresenta uma abordagem para diferentes modelos de *detecção de bordas* utilizando MapReduce. Para tanto, é utilizado um *cluster* de computadores heterogêneos em uma rede local com a plataforma Hadoop. Por se tratar de uma tarefa independente, a detecção de bordas é altamente distribuída, necessitando de uma etapa de redução apenas para unir os resultados de cada parte do processamento. Este estudo expõe o grande potencial para escalar o processamento de imagens de sensoriamento remoto e executar problemas geoespaciais complexos nesta plataforma.

Também utilizando *clusters* de computadores sobre o Hadoop, Surve e Paddune (2014) e Jin et al. (2014) apresentam versões da clusterização pelo método *K-means*. A segunda proposta é mais interessante, pois é voltada para a computação em nuvem. A estratégia adotada consiste na decomposição da imagem em recortes para o processamento independente, que é distribuído pelos elementos processadores disponíveis. Com o objetivo de eliminar os artefatos presentes no resultado final, uma etapa de união dos segmentos gerados baseada nas técnicas de janelas deslizantes é realizada. No caso, o pós-processamento é realizado em uma única máquina. Vale mencionar que uma imagem de 312.07 GB foi processada com sucesso nos experimentos realizados.

3.3.O desafio do crescimento de regiões distribuído

É possível verificar que uma estratégia comumente aplicada, tanto nas abordagens paralelas quanto nas distribuídas é a divisão da imagem em partições com o objetivo de dividir a carga de processamento. Na maioria dos casos, cada partição é processada de forma independente com a existência de algum tipo de comunicação entre os elementos processadores ou uma etapa posterior para agregar os resultados. Este tipo de processamento é realizado a fim de reduzir a

quantidade de artefatos gerados no resultado final da segmentação, o que afeta a qualidade e por consequência a utilidade do método em questão.

Também é possível notar na maioria dos trabalhos citados que o algoritmo de segmentação tem como princípio uma abordagem independente, que requer pouca dependência entre os segmentos; ou que utiliza um número pré-definido de sementes, o que também limita a dependência entre os mesmos.

Abordagens que levam em consideração todos os *pixels* como sementes, geralmente envolvem maior dependência entre os dados e geram maior complexidade para a definição do paralelismo. Outra questão que torna a paralelização ainda mais complexa é a utilização de critérios de homogeneidade envolvendo outros tipos de atributos além dos espectrais, como atributos morfológicos.

Este é o caso do algoritmo multiresolução apresentado em (Baatz e Schäpe, 2000): um algoritmo de segmentação de imagens baseado em crescimento de regiões, que considera todos os *pixels* como sementes e possui critérios de similaridades baseados em atributos espectrais e morfológicos. Este método foi escolhido nesta tese dada a sua complexidade e forte dependência de dados, de forma que ao desenvolver a distribuição do mesmo, é possível generalizar a solução para qualquer método de crescimento de regiões e ainda adaptá-la para diversas outras técnicas.

Cabe informar que este algoritmo tem sido amplamente utilizado pela comunidade de GEOBIA, em grande medida, graças à sua presença no popular *software* eCognition (eCognition, 2015). Uma versão deste também pode ser encontrada no *software* InteIMAGE (InterIMAGE, 2015). Para esboçar uma ideia de sua popularidade, na última conferência GEOBIA realizada em maio de 2014 (GEOBIA 2014, 2014), 18 trabalhos citam exatamente esta referência do algoritmo, sem contar outras referências a trabalhos posteriores que também possuem este algoritmo como base.

Estratégias paralelas para este algoritmo foram propostas em estudos anteriores a esta tese. Em (Happ et al., 2010) uma arquitetura com múltiplos núcleos é utilizada para processar recortes da imagem em *threads* diferentes. Nesta solução, o processamento dos segmentos localizados nas bordas dos recortes é postergado para uma etapa sequencial existente entre cada iteração paralela de crescimento de regiões. Já em (Happ et al., 2013) é apresentado um

algoritmo paralelo para execução em GPUs baseado na execução independente de cada *pixel*. A identificação dos melhores segmentos candidatos à fusão é realizada primeiramente dentro de um bloco de processamento e depois atualizada de forma global. Posteriormente a fusão dos segmentos ocorre em uma seção crítica.

As soluções supracitadas, no entanto, não estão preparadas para tratar as imagens grandes de altíssima resolução que vem sendo produzidas atualmente. Além disso, a quantidade de multiprocessadores existente nos computadores ou de elementos processadores existentes nas GPUs é limitada, o que restringe a capacidade de processamento disponível. A continuidade do trabalho se dá por intermédio do desenvolvimento de uma solução distribuída para *clusters* de computadores na nuvem.

Apesar dos grandes esforços na busca dentro da literatura atual, não se tem conhecimento de um método distribuído de segmentação de imagens baseada em crescimento de regiões que seja eficiente e escalável e que possa ser executado na nuvem. A forte dependência de dados existente neste tipo de e a dificuldade em tratar os artefatos gerados no processamento distribuído são alguns dos fatores que limitam este desenvolvimento.

Para ilustrar esta dificuldade, no último Simpósio Brasileiro de Sensoriamento Remoto (XVII SBSR, 2015) foi afirmado, durante a "Sessão Especial: Big Data em Observação da Terra: infraestruturas e análises espaço-temporais", que a segmentação de imagens por crescimento de regiões é tarefa impossível de ser realizada utilizando plataformas como o Hadoop. Esta afirmação, embora um pouco drástica, tem como base a dificuldade de se trabalhar com um método que além do processamento de dados *raster*, possui uma forte dependência espacial, em um ambiente distribuído que não leva em consideração a vizinhança dos objetos.

O método apresentado nesta tese visa justamente transpor esta barreira. O principal objetivo é fornecer uma solução capaz de realizar a segmentação de imagens de forma eficiente, robusta e escalável com a capacidade de tratar o grande volume de dados de sensoriamento remoto.

4
Segmentação de imagens distribuída

Neste capítulo, é apresentado o método proposto para a segmentação de imagens distribuída. A solução apresentada considera a técnica de crescimento de regiões e o modelo MapReduce para execução na nuvem a fim de processar imagens grandes de altíssima resolução.

Primeiramente, uma visão geral do método é apresentada. Em seguida, o ambiente distribuído e a representação dos dados são descritas de forma concisa. Na sequência, cada etapa do método é explanada de forma mais detalhada. Por fim, alguns detalhes relacionados com o desenvolvimento do método são apresentados.

4.1.Visão geral do método

De uma maneira geral o método pode ser descrito por intermédio de três etapas principais, conforme melhor descrito nas seções 4.4, 4.5 e 4.6, e ilustrado na Figura 4.1: 1) Divisão da imagem em *tiles*; 2) Segmentação independente de cada *tile;* 3) Pós-processamento do resultado inicial.

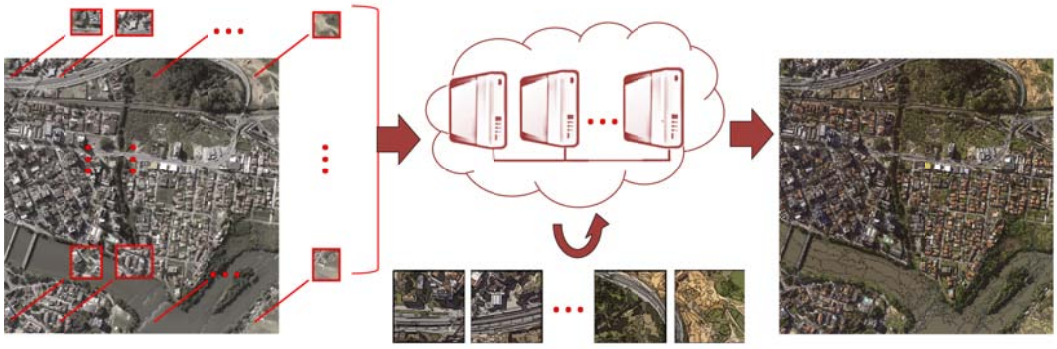


Figura 4.1.Esquema básico do método: 1) Imagem dividida em tiles; 2) Segmentação independente produzindo tiles com artefatos nas bordas; 3) Pós-processamento produzindo o resultado final.

A primeira etapa envolve a divisão da imagem em *tiles* a fim de gerar diferentes conjuntos de dados capazes de serem distribuídos durante o processamento na nuvem. Um sistema eficiente de indexação espacial hierárquico

é utilizado para identificar a vizinhança entre os *tiles*, conforme melhor explanado na seção 4.4.

A etapa seguinte consiste na execução distribuída do algoritmo de segmentação de imagens para cada *tile* gerado. O termo segmentação independente é utilizado, pois não há nenhum tipo de comunicação entre os processos. Por este motivo, o resultado desta etapa apresenta artefatos ao longo de toda a fronteira dos *tiles*.

A última etapa condiz com o pós-processamento dos segmentos gerados. Três métodos de pós-processamento são propostos nesta tese apresentando diferentes balanceamentos entre qualidade do resultado e velocidade de execução. É nesta etapa de pós-processamento que as ideias fundamentais para a segmentação distribuída são aplicadas.

O primeiro importante conceito está em assumir que os segmentos que não fazem fronteira com as bordas dos *tiles* já estão devidamente delineados. Embora esta suposição não seja necessariamente verídica para todos os casos, ela faz sentido em sua maioria, pois os segmentos em questão já foram analisados junto com todos os seus respectivos vizinhos. Esta definição permite que a etapa de pós-processamento envolva uma quantidade total de dados muito menor do que a inicial e permite que segmentos de diferentes *tiles* sejam agrupados em um mesmo processo sem que haja sobrecarga de memória.

A forma com que os segmentos são agrupados é outro conceito chave. Os segmentos gerados estão também associados a um índice espacial de tal forma que é possível facilmente agrupá-los de acordo com sua localização. Assim, segmentos próximos podem fazer parte do mesmo processo a fim de realizar um correto crescimento de regiões.

O terceiro conceito principal está na utilização dos níveis hierárquicos da indexação espacial para realizar um pós-processamento progressivo. Desta forma, é possível executar vários passos do pós-processamento considerando segmentos de uma quantidade de *tiles* cada vez maior em cada processo. Este fato garante que todos os segmentos existentes sejam analisados junto a todos os seus respectivos vizinhos.

O quarto e último conceito fundamental para o sucesso do método consiste na transformação dos segmentos gerados em *pixels,* antes de cada passo a ser realizado no pós-processamento. Isto faz com que os segmentos não fiquem

limitados ao crescimento da etapa anterior, promovendo a liberdade para que, ao se analisar uma vizinhança maior, o crescimento seja modificado de acordo com as novas informações.

4.2.Ambiente distribuído

O processamento distribuído se baseia nas plataformas Hadoop e Pig para o processamento MapReduce e a execução do fluxo de dados. A estratégia de segmentação é representada por um script Pig Latin que é carregado na nuvem. A plataforma Pig analisa este script e o compila em *jobs* MapReduce para realizar o processamento distribuído. Vale mencionar que o interpretador do Pig também é responsável por otimizar o script de modo a reduzir a quantidade e complexidade dos *jobs* MapReduce que são gerados.

Embora a compilação do script Pig Latin para os *jobs* MapReduce seja automática, tanto o Pig quanto o Hadoop fornecem uma série de parâmetros que permitem o correto ajuste dos sistemas. Inicialmente, a plataforma Pig lê alguns destes parâmetros de configuração presentes no script e depois carrega as bibliotecas necessárias de forma a enviá-las às máquinas responsáveis pela execução da segmentação.

A seguir, os *jobs* são iniciados e o Hadoop Mapreduce faz a leitura dos dados de entrada que estão presentes em um repositório auxiliar na nuvem. Os dados temporários produzidos durante o processamento da segmentação são armazenados no Hadoop Distributed File System (HDFS) e lidos quando necessários. Quando o resultado final é gerado, o mesmo é armazenado novamente no repositório auxiliar na nuvem.

É importante ressaltar que ao se optar por uma *engine*, como o Hadoop e o Pig, uma série de benefícios é obtida, como a utilização de métodos já consagrados e a simplificação do trabalho a ser realizado. Em contrapartida, perde-se flexibilidade para desenvolvimentos futuros e aprimoramentos específicos para cada caso.

4.3.Representação de dados

As imagens são os principais dados de entrada da segmentação distribuída. Inicialmente, a imagem é dividida em *tiles* que estão contidos em células

geográficas de acordo com um dado sistema de referência de coordenadas, conforme melhor explanado na seção 4.4. As células geográficas são representadas em uma hierarquia de diferentes níveis na qual o nível mais baixo corresponde ao nível dos *tiles*. É importante informar que um *tile* pode ser menor que as células devido à localização espacial do mesmo.

Cada *tile* é representado por dois arquivos: um é o recorte da imagem propriamente dita e o outro contém meta-informações que auxiliam no processamento do método como o número de *pixels* por linha e coluna da imagem, o número de bandas espectrais e as coordenadas geográficas do *tile* em questão.

Adicionalmente existe um arquivo que contém uma lista de todas as células geográficas existentes no nível do *tile*. Esta lista inclui a representação geoespacial das células no formato WKT (Well-Known Text) (Open Geospatial Consortium, 2015) e é utilizada para verificar a localização dos segmentos de acordo com sua posição nas células.

A segmentação distribuída produz segmentos representados na forma vetorial. Cada segmento carrega consigo dois tipos de informações: geometria e propriedades. A geometria corresponde à representação geoespacial do polígono pelo formato WKT. Já as propriedades representam atributos do vetor como o rótulo, localização da borda entre outros. Este tipo de dado é processado na forma de uma tupla e posteriormente armazenado no formato JSON (JavaScript Object Notation) de forma a ser devidamente processado pelo Pig quando necessário. JSON é um formato leve baseado em texto bastante utilizado para troca de dados. Além de ser independente de linguagem, também é de fácil entendimento para humanos e para máquinas (JSON, 2015).

4.4.Divisão da imagem

Dividir a imagem em recortes de tamanhos idênticos não é novidade quando se trata da segmentação de imagens paralela ou distribuída. A forma como isto é realizado, no entanto, pode apresentar algumas variações.

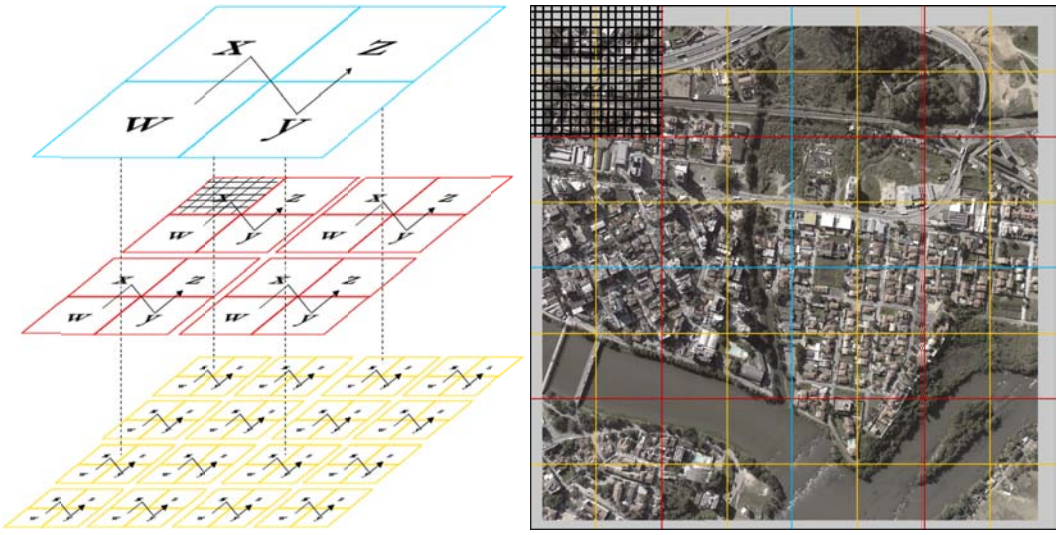
Neste trabalho, a divisão da imagem é baseada em um dado sistema de referência de coordenadas (SRC) e é, portanto, independente dos limites das imagens. Esta escolha provê maior robustez em relação à qualidade do resultado

final, pois mantém a mesma referência de origem quando diferentes processos de segmentação são necessários. Isto é importante quando se trabalha com séries multitemporais, imagens de diferentes resoluções, imagens com variações de tamanho, etc. Além disso, esta característica favorece a utilização do método dentro de sistemas de análise de imagens, onde múltiplas imagens geralmente são consideradas.

A divisão da imagem é realizada de forma hierárquica, através da definição de camadas de células geográficas, tal como demonstrado na Figura 4.2. A camada de nível superior engloba toda a região definida no sistema de coordenadas. Cada célula é então dividida, de forma recursiva, em quatro outras células na camada inferior seguinte até que se alcance o tamanho desejado, construindo assim uma estrutura semelhante a uma *quadtree*.

Para cada célula geográfica é atribuído um rótulo único que carrega a informação a respeito da sua posição na estrutura hierárquica utilizando uma curva de preenchimento do espaço (*space-filling curve* - Sagan, 1994), mais especificamente uma curva de ordem Z (*Z-order curve*). Esta curva permite mapear um espaço multidimensional para uma única dimensão enquanto preserva a localidade. Assim, incluir ou eliminar um caractere de um rótulo corresponde a mover através dos níveis hierárquicos. Tomemos, por exemplo, a célula com hachuras na Figura 4.2(a) rotulada como XX. Esta célula faz parte da célula X no nível superior e é decomposta nas células XXW, XXX, XXY e XXZ no próximo nível inferior.

Os *tiles* de imagem gerados estão contidos nas células geográficas presentes no nível mais baixo. Assim, o tamanho das células geográficas é fixo e pré-determinado enquanto que o tamanho dos *tiles*, apesar de definido, pode variar de acordo com a localização nas bordas da imagem - Figura 4.2-b. Os *tiles* são salvos em arquivos de imagem que acompanham arquivos com meta informações acerca de suas coordenadas, tamanhos, número de bandas, etc. Estes arquivos são carregados em um repositório na nuvem juntamente com um arquivo adicional contendo todas as células geográficas e suas coordenadas.



(a) (b)

Figura 4.2. Esquema de divisão da imagem: (a) curva de ordem Z e células geográficas compondo uma quadtree; (b) imagem dívida em 64 tiles contendo células geográficas de diferentes níveis (diferentes cores).

A definição do tamanho das células geográficas, e consequentemente dos *tiles*, é uma questão que deve ser discutida. Se extrapolarmos para os limites, um *tile* pode ser tão pequeno quanto um *pixel* ou tão grande que envolva a imagem inteira. Neste sentido, tamanhos menores acarretam em um maior número de *tiles*, o que pode favorecer inicialmente a distribuição dos dados em um maior número de elementos processadores. Já tamanhos maiores implicam em uma menor área de fronteira, o que reduz o custo do pós-processamento. Deve ser considerado que tamanhos "muito pequenos" incidem em um alto custo adicional de processamento (*overhead*) e podem deteriorar a qualidade final da segmentação, enquanto que tamanhos "muito grandes" reduzem a escalabilidade e podem causar problemas em relação à capacidade de memória.

Vale ainda mencionar que na implementação do método proposto a divisão da imagem ocorre localmente, de forma a que se realize o *upload* dos diversos arquivos já divididos. Além disso, para uma mesma divisão da imagem, não é necessário realizar o processo novamente podendo utilizar os arquivos já armazenados na nuvem.

4.5.Segmentação de imagens independente

Uma vez que a imagem tenha sido dividida é possível executar o algoritmo de segmentação para cada *tile* da imagem de forma distribuída. Esta etapa se

assemelha bastante com uma execução sequencial de diversas imagens distintas em diversos computadores. A principal diferença é a estrutura de execução em nuvem. Ao final do processo, os segmentos gerados são vetorizados, i.e., transformados em polígonos, de forma a facilitar qualquer processamento futuro. Atributos que podem ser úteis posteriormente como o rótulo relacionado à célula geográfica em que ele está localizado, atributos espectrais, localização em relação à borda, etc. podem ser incluídos conforme a necessidade.

A segmentação poderia terminar neste ponto, armazenando os resultados gerados no repositório na nuvem. Porém, o resultado das segmentações independentes para crescimento de regiões apresenta artefatos ao longo de todas as fronteiras dos *tiles* (Figura 4.3). Isto ocorre pelo fato de não haver qualquer tipo de comunicação entre os processos a fim de acelerar o processamento e permitir que esta etapa seja completamente independente e escalável. Vale mencionar que para algumas técnicas de segmentação, por exemplo, limiarizações, esta abordagem já seria suficiente para produzir o resultado esperado.

Com o intuito de reduzir e/ou eliminar os artefatos presentes no resultado, uma etapa de pós-processamento deve ser considerada. Esta etapa é descrita a seguir.

Figura 4.3. Parte do resultado da segmentação independente com artefatos presentes ao longo de todas as bordas dos *tiles*.

4.6.Pós-processamento

Como os problemas da segmentação estão localizados nas bordas dos *tiles* da imagem é possível considerar que todos os segmentos internos ao *tile,* ou seja, os segmentos que não tocam as fronteiras dos *tiles*, estão devidamente formados.

Desta maneira, é possível filtrar todos os segmentos internos e considerá-los como definitivos, armazenando-os no repositório como dados de saída. Enquanto isto, os segmentos que tocam as fronteiras dos *tiles* devem ser submetidos a um pós-processamento na tentativa de realizar novas fusões e eliminar os artefatos existentes.

É importante salientar que o pós-processamento tem que ser robusto e eficiente. Isto significa dizer que deve haver uma distribuição inteligente dos segmentos localizados nas bordas a fim de que não haja problemas de capacidade de memória e de que seja possível obter resultados com qualidade satisfatória.

Três variantes de pós-processamento são propostas e avaliadas nesta tese apresentando diferentes balanços entre qualidade do resultado e velocidade de execução. A primeira e mais simples consiste em um único passo, fornecendo uma nova chance de união para segmentos em *tiles* adjacentes. Este método tende a ser o mais rápido, porém tende também a apresentar um maior número de artefatos ao final do processo. A segunda alternativa envolve um pós-processamento iterativo percorrendo os diferentes níveis da *quadtree* formada pelas células geográficas. Um maior tempo de processamento é necessário, porém a quantidade de artefatos resultantes é bastante reduzida. A terceira e última variante se assemelha com a anterior, mas envolve uma ressegmentação a cada nível da *quadtree* executado. Esta solução apresenta um resultado sem artefatos, mas tende a apresentar um maior tempo de execução.

4.6.1.Pós-processamento simples

Segmentos internos que não fazem contato com qualquer borda dos *tiles* da imagem são considerados como definitivos e podem ser salvos no repositório de dados na nuvem. Os segmentos restantes são então agrupados de forma a gerar uma nova massa de dados distribuída para ser executada no *cluster*.

A ideia é criar uma quantidade de grupos igual à quantidade de *tiles* a fim de obter uma carga similar à etapa de segmentação independente. Desta forma, cada grupo de segmentos é associado a um *tile* de imagem original.

O agrupamento dos segmentos deve tentar reunir os segmentos adjacentes localizados em diferentes *tiles* de forma a possibilitar a fusão dos mesmos. A Figura 4.4 ilustra estes grupos através de diferentes cores. Os segmentos esverdeados são aqueles que não tocam as bordas e não são submetidos para o pós-processamento. Note que cada grupo contém segmentos de diferentes *tiles*.

Para definir o grupo de um determinado segmento verificam-se todos os *tiles* que fazem fronteira com o mesmo, incluindo o *tile* ao qual ele pertence. A seguir, de acordo com um determinado critério, seleciona-se um dentre estes *tiles* para designar o grupo deste segmento.

Para obter uma coerência global, o critério de seleção do *tile* de fronteira utilizado nesta tese foi por intermédio do rótulo da célula geográfica em que o *tile* está contido: o segmento será incorporado ao grupo que está associado ao *tile/*célula de menor rótulo, ou seja, àquele localizado mais abaixo e mais à esquerda do segmento. Para um melhor entendimento é possível usar a Figura 4.4 como exemplo. Neste caso, considera-se que cada *tile* possui um rótulo numérico como identificado na figura. Assim, para cada segmento, verificam-se todos os *tiles* que o tocam e escolhe-se aquele com menor rótulo. Neste caso, todos os segmentos de borda que estão em contato com o *tile* 1, o menor de todos, são agrupados em rosa na figura.

Cada grupo de segmentos gerado é então submetido a uma unidade de processamento para realizar a etapa de pós-processamento. Neste caso, diversas estratégias podem ser adotadas para decidir se dois segmentos adjacentes devem ser fundidos ou não. A decisão pode ser baseada em um simples critério de similaridade ou consistir na execução de uma nova sequência de iterações de crescimento de regiões.

Esta estratégia de pós-processamento consiste de uma única etapa, tratando-se, portanto, de uma solução de execução rápida. No entanto, ela não processa corretamente segmentos adjacentes pertencentes a diferentes grupos, assim como não trata segmentos que poderiam expandir para além de mais de dois *tiles* adjacentes como é o caso dos segmentos em destaque na Figura 4.4. Assim, certa

quantidade de artefatos nas bordas dos *tiles* pode ser observada no resultado final da segmentação.

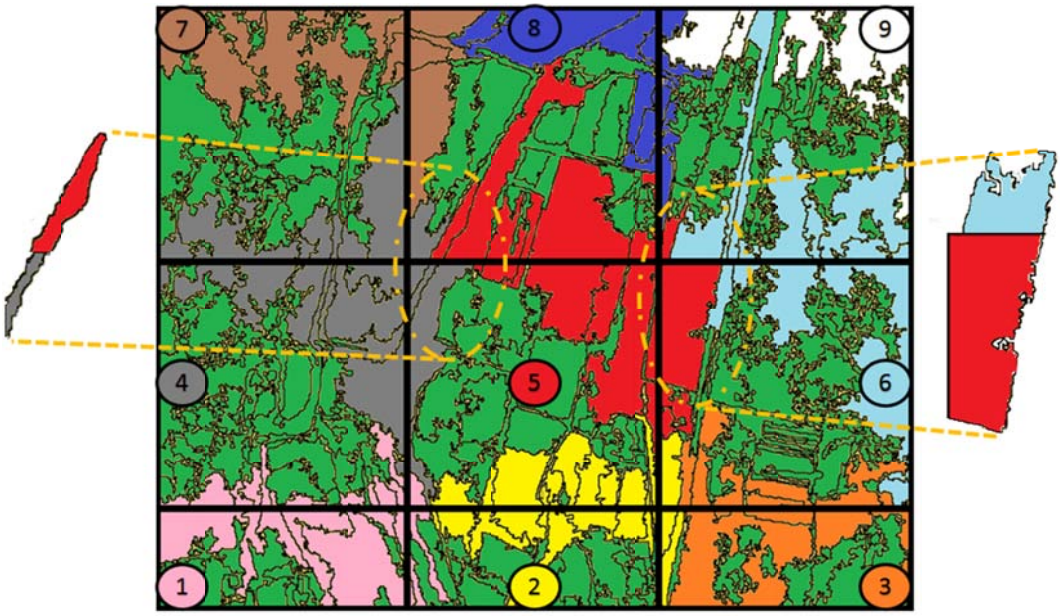


Figura 4.4. Segmentos agrupados após a segmentação independente. Segmentos em verde não fazem contato com as bordas. Diferentes cores ilustram diferentes grupos. Em destaque: segmentos em contato que fazem parte de diferentes grupos.

Para lidar com esses problemas, uma abordagem alternativa, de pós-processamento hierárquico, é proposta. Esta estratégia, descrita a seguir, envolve sucessivas etapas de redução agrupando as células geográficas através dos níveis da *quadtree*, até que a imagem inteira seja considerada.

4.6.2.Pós-processamento hierárquico

O pós-processamento hierárquico é uma estratégia iterativa que realiza o processamento em fases a fim de garantir que todos os segmentos tenham a possibilidade de fundir com todos seus segmentos adjacentes. Como o nível mais baixo da *quadtree* está relacionado com o próprio *tile*, a primeira fase tem início no nível seguinte. As outras fases são adicionadas até que se atinja um nível grande o suficiente para englobar a imagem toda.

O processamento dentro de cada fase é exatamente o mesmo, variando apenas os dados de entrada, que são sempre definidos ao final da fase anterior. Tendo como base o nível atual da *quadtree*, os segmentos internos às células geográficas são considerados como definitivos e armazenados no repositório na

nuvem. Segmentos que fazem fronteira com segmentos de outra célula geográfica no nível atual e que também fazem parte da mesma célula geográfica no nível seguinte, são agrupados para serem processados juntos na próxima fase. Os segmentos restantes são postergados para as próximas fases. As Figuras 4.5 e 4.6 exemplificam duas etapas do pós-processamento hierárquico, onde os segmentos em verde são definitivos, em branco são postergados para a fase seguinte e os coloridos fazem partes do mesmo agrupamento. As células de diferentes níveis são representadas pelas linhas de grade em preto e dourado (Figura 4.5); e em preto, dourado e vermelho (Figura 4.6).

O agrupamento dos segmentos ocorre de acordo com a posição da sua célula geográfica no próximo nível hierárquico. Como descrito anteriormente (seção 4.2), isso pode ser realizado removendo a última letra do rótulo relacionado à célula geográfica do nível atual (p. ex. XWXZ → XWX).

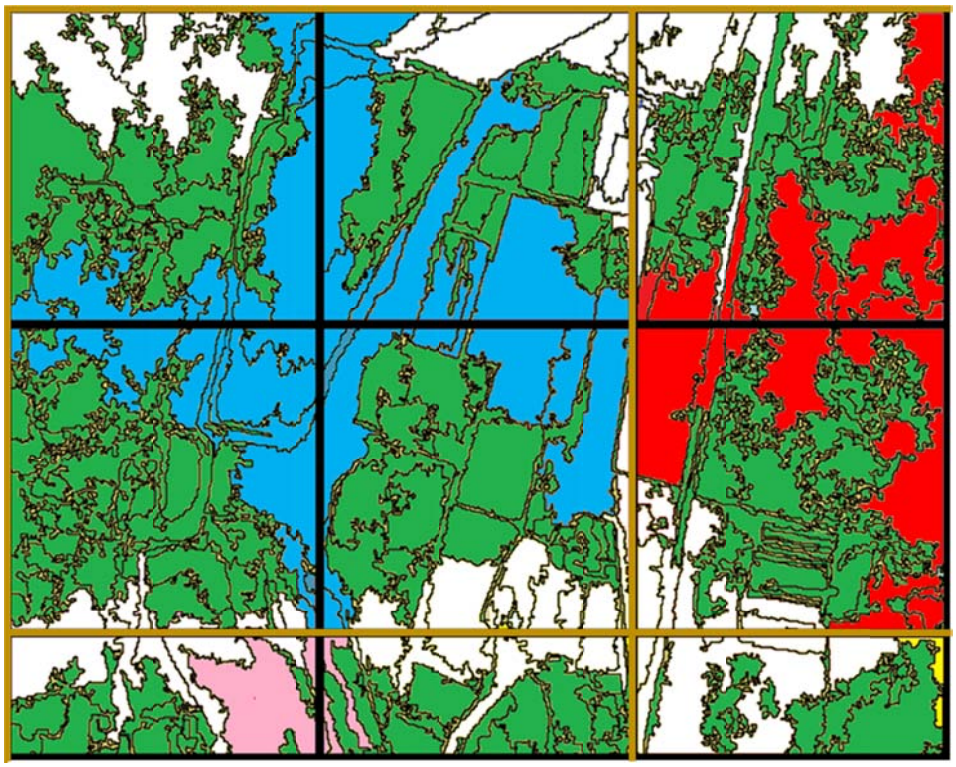


Figura 4.5. Exemplo de segmentos agrupados para a primeira fase do pós-processamento hierárquico. Segmentos em verde não fazem contato com as bordas, segmentos em branco são postergados para as fases seguintes e segmentos coloridos fazem parte do mesmo agrupamento. Linhas em preto e dourado definem as fronteiras das células em diferentes níveis hierárquicos.

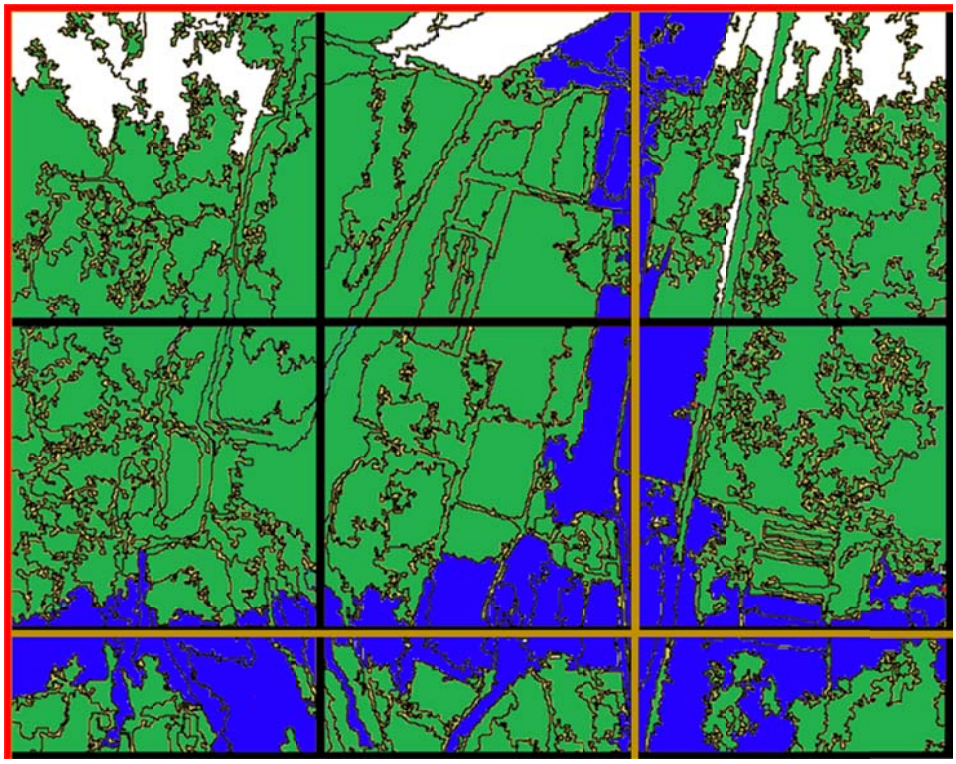


Figura 4.6. Exemplo de segmentos agrupados para a segunda fase do pós-processamento hierárquico. Segmentos em verde não fazem contato com as bordas, segmentos em branco são postergados para as fase seguintes e segmentos coloridos fazem parte do mesmo agrupamento. Linhas em preto, dourado e vermelho definem as fronteiras das células em diferentes níveis hierárquicos.

É importante informar que os segmentos que tocam a borda externa das células são sempre carregados para as fases seguintes. Quando um destes segmentos também faz parte de um agrupamento significa que ele será processado em diferentes níveis (a comparação entre as Figuras 4.5 e 4.6 permite identificar alguns casos). Isto é o que garante a redução dos artefatos no resultado da segmentação, uma vez que existe a verificação de toda a vizinhança daquele segmento, independente de sua localização na fronteira. Os segmentos que simplesmente são postergados visam restringir a quantidade de dados processada em cada fase reduzindo os custos de processamento e utilização da memória, uma vez que não há necessidade de processá-los naquela fase.

A definição do nível hierárquico de cada segmento é realizada inicialmente na etapa de segmentação independente. Esta informação é armazenada na forma de atributo do segmento. Ao término de cada etapa de pós-processamento uma nova verificação é realizada, de forma a identificar os segmentos que farão parte do processamento de níveis superiores e os que serão armazenados.

A identificação do nível de processamento é baseada na localização do segmento em relação às bordas das células geográficas dos diferentes níveis. Na Figura 4.7 é possível identificar os segmentos que foram separados nos diferentes níveis de processamento. Na verdade, a Figura 4.7 ilustra os resultados incrementais após a segmentação independente de cada fase do pós-processamento. A sequência de resultados pode ser visualizada de (a) a (f) e cada cor representa o resultado de um nível/fase diferente. Neste exemplo a imagem a ser segmentada está contida em mais de uma célula geográfica. A Figura 4.7(a) apresenta apenas os segmentos internos, i.e., aqueles que não tocam a borda dos *tiles*, após a etapa de segmentação independente. O resultado da primeira fase de pós-processamento é mostrado na Figura 4.7(b), onde os segmentos internos às células geográficas do segundo nível hierárquico são armazenados no repositório na nuvem. As demais figuras representam os segmentos processados desde a segunda fase (c) até a última fase de pós-processamento (f).

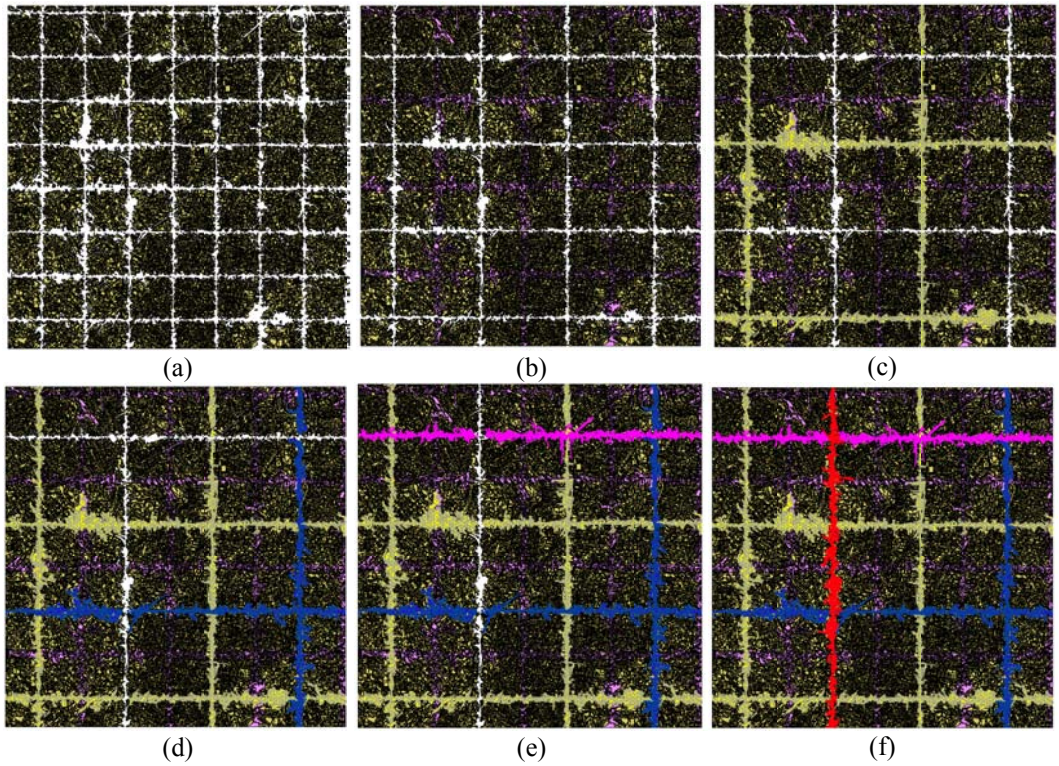


Figura 4.7. Resultado da segmentação após diferentes fases. (a) segmentos internos após segmentação independente. De (b) a (f): resultado incremental após cada fase em diferentes cores.

Em cada fase, o número de grupos de segmentos (e de processos) equivale ao número de células geográficas existentes naquele nível hierárquico. Desta

forma, a eficiência computacional tende a cair quando o processamento atinge os níveis mais altos, pois há uma menor quantidade de células. Outra questão envolvendo os níveis mais altos está relacionada com a memória, já que muitos objetos podem ser agrupados em uma mesma máquina.

Os resultados produzidos pela etapa de pós-processamento hierárquico reduzem consideravelmente a presença de artefatos na imagem, porém ainda é possível verificar a presença de alguns nas bordas dos *tiles*. Isto ocorre pelo fato do processamento ser realizado sobre segmentos já crescidos anteriormente, o que limita as condições para fusão, i.e., o atendimento do critério de homogeneidade associado ao algoritmo de segmentação. A solução para este problema envolve a dissolução destes segmentos novamente em *pixels* conforme descrito na seção seguinte.

4.6.3.Pós-processamento hierárquico com ressegmentação

O funcionamento do pós-processamento hierárquico com ressegmentação possui o mesmo princípio do pós-processamento hierárquico descrito na seção anterior (seção 4.6.2), podendo também ser exemplificado pelas Figuras 4.5 e 4.6. A diferença principal é que no início de cada fase de pós-processamento todos os segmentos agrupados para serem processados são novamente dissolvidos em *pixel* e uma nova segmentação de crescimento de regiões é realizada sobre a área da imagem.

Esta ressegmentação é o que faz com que os segmentos tenham liberdade para crescer de diferentes maneiras, analisando novamente toda a vizinhança e eliminando os artefatos no resultado final.

Como neste momento todos os segmentos são representados por vetores, a dissolução em *pixel* envolve um processo de rasterização destes polígonos. Para tanto, é necessário acessar novamente os *tiles* de imagem presentes no repositório de dados. Este processo deve ser realizado de forma inteligente a fim de manter a eficiência do método.

Vale lembrar ainda que, conforme as fases avançam, segmentos podem fazer parte de mais de um *tile* sendo necessário um tratamento para isto. Também é importante realizar um mapeamento dos *pixels* considerando a imagem como

um todo para que as vizinhanças sejam definidas corretamente e seja possível a execução do processo de crescimento de regiões de forma integral.

Esta solução é a que envolve maior tempo de processamento e também maior uso da memória. No entanto, ela fornece uma nova chance de crescimento para os segmentos e por consequência produz um resultado sem a presença de artefatos.

4.7.Detalhes de desenvolvimento

Um programa foi desenvolvido para realizar a divisão da imagem de acordo com o sistema de referência de coordenadas e o tamanho escolhido pelo usuário. Adicionalmente é possível realizar o carregamento dos arquivos diretamente no repositório na nuvem.

Os algoritmos de segmentação e pós-processamento, que vão operar sobre os dados de entrada, foram desenvolvidos como UDFs Pig (User Defined Functions) em Java com o objetivo de serem invocadas pelo script Pig Latin. Este script é o verdadeiro responsável por definir o fluxo de processamento do método e também por criar os devidos *jobs* MapReduce. Um programa foi criado para criar um *script* PIG de forma automática de acordo com um *template* e com a definição de alguns parâmetros pelo usuário.

O script Pig define inicialmente o carregamento dos dados de entrada e o devido balanceamento de carga para que, a segunda etapa, condizente com a segmentação independente, seja executada. A UDF de segmentação independente é executada como um Map para cada conjunto de *tiles* alocado em cada elemento de processamento. Estes passos são realizados por intermédio de operações LOAD, para carregar os dados, e FOREACH, para processar cada elemento carregado.

Como resultado, os segmentos gerados são representados por polígonos contendo alguns atributos importantes como o rótulo da célula geográfica associada e se o segmento é interno ou faz parte da borda do *tile*. Assim, é necessária uma etapa de vetorização ao final do processamento da UDF de segmentação.

A segmentação independente também fornece, para cada segmento gerado, um rótulo que está relacionado de alguma forma com as células geográficas

existentes. É importante ressaltar que a definição deste rótulo depende do tipo de pós-processamento a ser realizado, como descrito em mais detalhes na seção 4.6. De uma forma geral, para o pós-processamento simples, o rótulo corresponde a um dos *tiles* que faz fronteira com o segmento em questão, enquanto nos pós-processamentos hierárquicos ele corresponde exatamente à célula geográfica ao qual o segmento está contido.

Para a devida separação entre os segmentos internos e os outros segmentos de borda gerados, utiliza-se a instrução SPLIT, que divide os dados de acordo com um determinado critério. No caso do pós-processamento hierárquico, os segmentos são também divididos entre os diversos níveis de processamento existentes. Neste momento, os segmentos internos, i.e., que não tocam as bordas dos *tiles* da imagem, podem ser armazenados no repositório na nuvem, reduzindo a quantidade de dados existente durante o pós-processamento. A função STORE é utilizada para este fim.

Os segmentos restantes são agrupados de acordo com o rótulo associado a eles por meio de uma instrução COGROUP BY, responsável por agrupar diferentes conjuntos de dados. É importante notar que nos casos de pós-processamento hierárquico o agrupamento é feito de acordo com o próximo nível hierárquico, e assim, utiliza-se somente parte dos rótulos existentes. Uma UDF foi desenvolvida para retornar apenas a parte do rótulo desejada (SUBSTRING).

Para cada grupo existente, a UDF correspondente ao pós-processamento desejado é invocada por uma cláusula FOREACH, dando início a terceira etapa: o processamento distribuído dos grupos por meio de Reduce. No caso do pós-processamento simples, o resultado gerado é armazenado no repositório e a segmentação é dada como encerrada. No caso dos pós-processamentos hierárquicos uma nova separação entre os segmentos de borda e os segmentos internos é realizada considerando os novos limites de acordo com o nível hierárquico atual. Da mesma forma, o agrupamento é realizado segundo este novo nível e mais uma fase de pós-processamento é realizada. O processo se repete até que não haja mais segmentos de borda no nível hierárquico, ou seja, o número de iterações é definido de acordo com o nível que contém a imagem inteira.

5
Análise experimental

Com o objetivo de validar o método apresentado, experimentos foram executados utilizando um *cluster* virtual em uma infraestrutura de nuvem comercial. A implementação é baseada no algoritmo de crescimento de regiões multiresolução com heurística de melhor vizinho local mútuo para fusão (Baatz e Schäpe, 2000), descrito na Seção 2.2, e utiliza a plataforma Pig para gerar os *jobs* MapReduce que serão executados no Hadoop. O fluxo de dados distribuído foi programado em Pig Latin e as funções de segmentação e pós-processamento foram escritas como UDFs em Java. Um programa para escrever automaticamente os scripts Pig também foi desenvolvido, bem como ferramentas adicionais para divisão da imagem, *upload* dos dados na nuvem, etc.

Inicialmente, se sucedeu um breve estudo acerca dos tamanhos das células de nível inferior e, por consequência, dos *tiles*. Em seguida dois tipos de experimentos foram realizados: um para verificar o potencial de escalabilidade e eficiência do método e outro para analisar a qualidade do resultado da segmentação.

5.1. Ambiente de execução

O ambiente de execução consiste numa combinação de serviços na nuvem: um repositório para armazenar os dados de entrada e de saída, bem como os programas e bibliotecas necessárias; e um serviço que fornece *clusters* Hadoop para executar a segmentação de imagens distribuída.

Neste trabalho foram utilizados serviços oferecidos pela AWS (Amazon Web Services): os processamentos foram executados no EMR (Amazon Elastic MapReduce) e os dados armazenados no S3 (Amazon Simple Storage Service). O EMR possibilita a definição da quantidade e do tipo de máquinas a ser utilizado. No caso, máquinas do tipo *m3.xlarge*, com processador Intel Xeon E-5-2670 v2 operando a 2.5GHz, arquitetura de 64 bits, 4 núcleos físicos (8 lógicos), 15 GB de RAM e dois discos de 40 GB com tecnologia SSD, foram utilizadas. Vale

mencionar que as máquinas selecionadas possuem Hadoop 2.4.0 e Pig 0.12 instalados. Uma descrição mais detalhada destes serviços pode ser encontrada em (Amazon Web Services, 2015-b).

Algumas definições podem ser configuradas de acordo com a necessidade. O fator de replicação de dados do HDFS depende da quantidade de máquinas existentes no *cluster*: 3 para 10 ou mais nós, 2 para 4 a 9 nós e 1 para 3 ou menos nós. O tipo de compressão de dados foi definido para Snappy (Snappy, 2015) e o limite de contadores do Hadoop foi aumentado entre outras alterações.

A fim de reduzir os gastos com o provisionamento do *cluster*, é possível reservar instâncias do tipo *spot* no EC2 (Amazon Elastic Compute Cloud) pelo EMR. Os preços para utilização destas máquinas flutuam de acordo com a demanda e a oferta e geralmente são muito mais baixos do que as instâncias reservadas na modalidade *on demand*. Por exemplo, o custo atual de utilização de uma instância (máquina) do tipo m3.xlarge, utilizada nos experimentos, é de 0,266 dólares por hora quando reservada na modalidade *on demand*. Este valor é reduzido para algo entre 0,035 a 0,041 dólares por hora ao utilizar uma instância na modalidade *spot*. A desvantagem de usar instâncias *spot* é que acarretam em um maior tempo na inicialização do *cluster,* além disso, elas podem ser finalizadas pelo provedor caso o valor de mercado sofra um grande aumento.

Em relação ao processamento, o procedimento de segmentação é feito da seguinte forma: a imagem é dividida em *tiles* que são armazenados no Amazon S3. Neste mesmo local estão armazenados o script Pig Latin, as UDFs e as bibliotecas necessárias para a execução da segmentação distribuída. As máquinas do EMR são preparadas e o *script* Pig é lido e traduzido em *jobs* MapReduce que são executados de forma distribuída. Ao longo do processamento, os resultados são armazenados de volta ao repositório existente no S3.

5.2.Avaliação de desempenho computacional

O objetivo destes experimentos é avaliar e comparar o desempenho computacional do método proposto. A ideia é verificar o *speedup*, a eficiência e a escalabilidade para cada uma das três soluções de pós-processamento propostas de acordo com uma quantidade crescente de máquinas utilizadas para processar imagens de dimensões cada vez maiores.

É importante ressaltar que uma máquina é sempre reservada para o Hadoop *JobTracker*, que é responsável por escalonar e gerenciar as tarefas e, portanto, não está disponível para realizar a segmentação. Assim, além daquela máquina, foram utilizadas 1, 2, 8, 16 e 32 máquinas do tipo *m3.xlarge*. Como cada máquina possui 8 núcleos lógicos, os experimentos foram realizados com 8, 16, 64, 128 e 256 unidades de processamento.

5.2.1.Base de Dados

Para os experimentos foi utilizado um recorte de 4000 x 4000 *pixels* de uma cena do satélite Quickbird, denominada 4k (Figura 5.1), que compreende alguns bairros da cidade de São Paulo. A imagem possui 4 bandas espectrais (azul, verde, vermelho e infravermelho) e resolução de 0,61 metros.

Figura 5.1. Imagem de teste 4k.

A imagem em questão foi replicada a fim de se obter imagens maiores para verificação do desempenho do método. Foram geradas imagens de 8000 x 8000 *pixels* (8k), 16000 x 16000 *pixels* (8k) e 32000 x 32000 *pixels* (32k), cujos tamanhos. Vale salientar que a replicação da imagem pode vir a ocultar a complexidade da segmentação em certas áreas da imagem. Porém, dentro do escopo desta tese, esta operação cumpre o objetivo de analisar grandes volumes dados.

Para realizar a divisão das imagens, *tiles* de 1024 x 1024 *pixels* foram considerados. Esta escolha foi realizada com base em experimentos preliminares que demonstraram que este tamanho produzia melhor desempenho computacional e maior qualidade na segmentação. Uma breve análise é apresentada na seção 5.2.2. Os parâmetros de segmentação utilizados (Tabela 5.1) foram baseados em (Novack, 2010) onde a mesma imagem foi utilizada em uma classificação de uso e cobertura do solo.

Escala	Cor / forma	Compacidade / Suavidade	Peso das Bandas
41	0,84	0,8	1,1,1,1

Tabela 5.1: Conjunto de parâmetros utilizado para segmentação.

5.2.2. Definição do Tamanho dos Tiles

Um breve estudo foi realizado com a finalidade de definir o tamanho dos *tiles*. Vale mencionar que uma análise mais ampla deve ser considerada em pesquisas futuras.

O uso da mesma dimensão horizontal e vertical, bem como o de potências de dois é comum neste tipo de divisão em tiles e facilitam os cálculos durante o processo (Sample e Ioup, 2010). *Tiles* de 256 x 256 *pixels* são pequenos demais para assegurar a qualidade da segmentação, enquanto *tiles* de 2048 x 2048 apresentaram problemas de memória durante os experimentos. Desta forma, apenas dois tamanhos foram considerados neste trabalho: 512 x 512 pixels e 1024 x 1024 *pixels*.

Como o estudo visa apenas à determinação do tamanho de *tile* para os experimentos seguintes e não pretende realizar uma análise exaustiva, os testes

foram limitados às imagens 4k e 32k, aos métodos de pós-processamento simples e hierárquico e ao uso de 1 e 16 nós. Os resultados são apresentados na Figura 5.2.

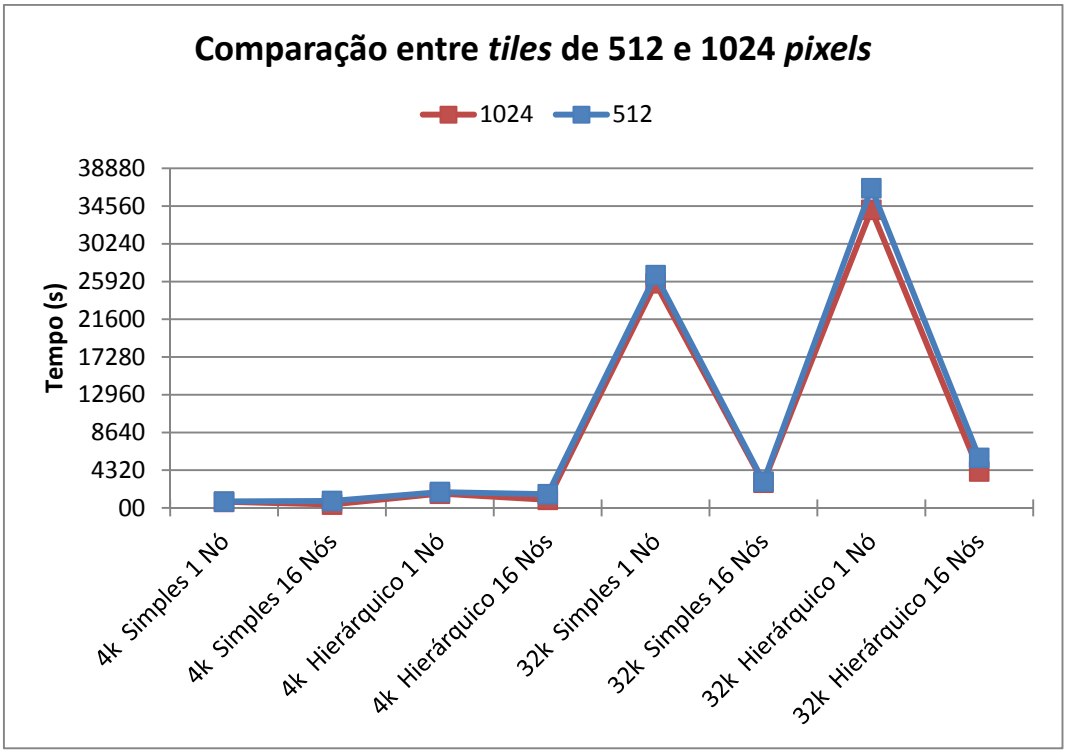


Figura 5.2. Comparação do tempo de execução para tiles de 512x512 e 1024x1024 pixels.

Nota-se que, apesar dos tempos de execução não serem muito diferentes, o uso de *tiles* de tamanho 1024x1024 *pixels* alcançam melhores resultados (menores tempos de execução) em todos os casos. Este fato, aliado com a premissa de que quanto maior o tamanho do *tile* menor é a interferência na qualidade do resultado, fez com que este seja o tamanho definido como padrão nos demais experimentos desta tese.

5.2.3.Resultados e Discussões

Os experimentos relacionados ao desempenho computacional foram realizados considerando os três métodos apresentados e variando os tamanhos de imagem e o número de máquinas de processamento.

É importante informar que a execução do método com pós-processamento hierárquico com ressegmentação utilizando apenas 1 e 2 máquinas para a imagem 32k resultou em erros e, portanto, não está representada nos gráficos desta tese. Este fato está relacionado à baixa distribuição do grande volume de dados

envolvido e a problemas no gerenciamento de memória nestes casos. Ao aumentar o número de nós de execução, o problema foi resolvido, como pode ser verificado nos resultados apresentados. Uma reestruturação, em termos de programação, envolvendo melhor utilização da memória pela estrutura de dados pode também resolver este problema.

O tempo de processamento para as imagens 4k, 8k, 16k e 32k é exibido nas Figuras 5.3, 5.4, 5.5 e 5.6, respectivamente. Os mesmos resultados são apresentados sob outro ponto de vista nas Figuras 5.7, 5.8 e 5.9, onde o tempo de processamento é exibido para os métodos de pós-processamento simples, hierárquico e hierárquico com ressegmentação.

Por meio dos gráficos, é possível verificar que o tempo de processamento diminui conforme o número de máquinas aumenta. Este comportamento é esperado, pois um maior poder computacional tende a resultar em menores tempos de execução. Esta redução é consideravelmente maior quando o tamanho da imagem aumenta, pois é possível se beneficiar da maior quantidade de unidades de processamento e, assim, alcançar um maior nível de processamento distribuído. No caso das imagens menores, algumas unidades de processamento ficam inativas, o que faz com que o paralelismo não seja explorado ao máximo. Estes fatos demonstram o potencial escalável do método e indicam que este potencial está também relacionado ao tamanho das imagens de entrada.

Ainda por meios destes gráficos é possível comparar a velocidade de execução de cada um dos métodos de pós-processamento. O pós-processamento simples resulta em menor tempo de processamento, enquanto o pós-processamento hierárquico com ressegmentação é o que obtém o maior tempo de execução, apesar de se aproximar do hierárquico em alguns casos. De fato, o pós-processamento simples é o que envolve o menor custo de processamento, uma vez que o hierárquico e o hierárquico com ressegmentação necessitam de diversas etapas progressivas de processamento contemplando um maior número de *jobs*. Esses últimos são idênticos nesse quesito, se diferenciando apenas na necessidade de realizar a rasterização e a ressegmentação dos segmentos gerados anteriormente.

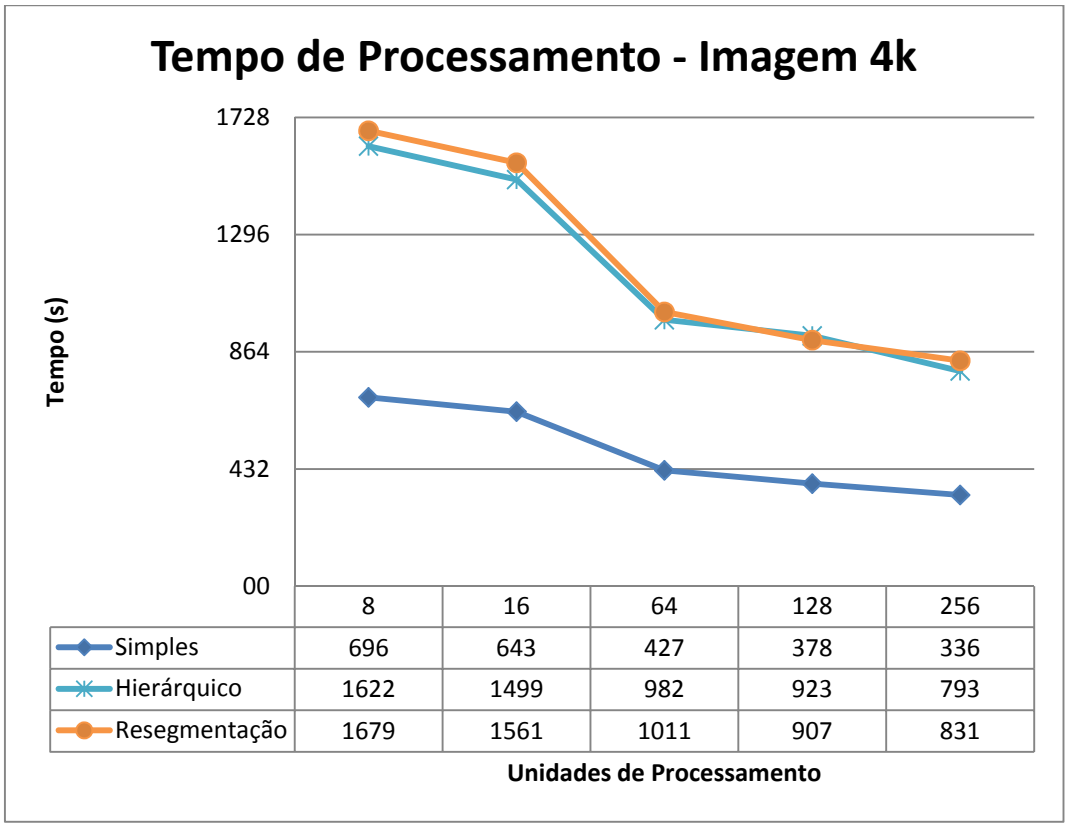


Figura 5.3. Tempo de processamento da Imagem 4k para cada método variando o número de máquinas.

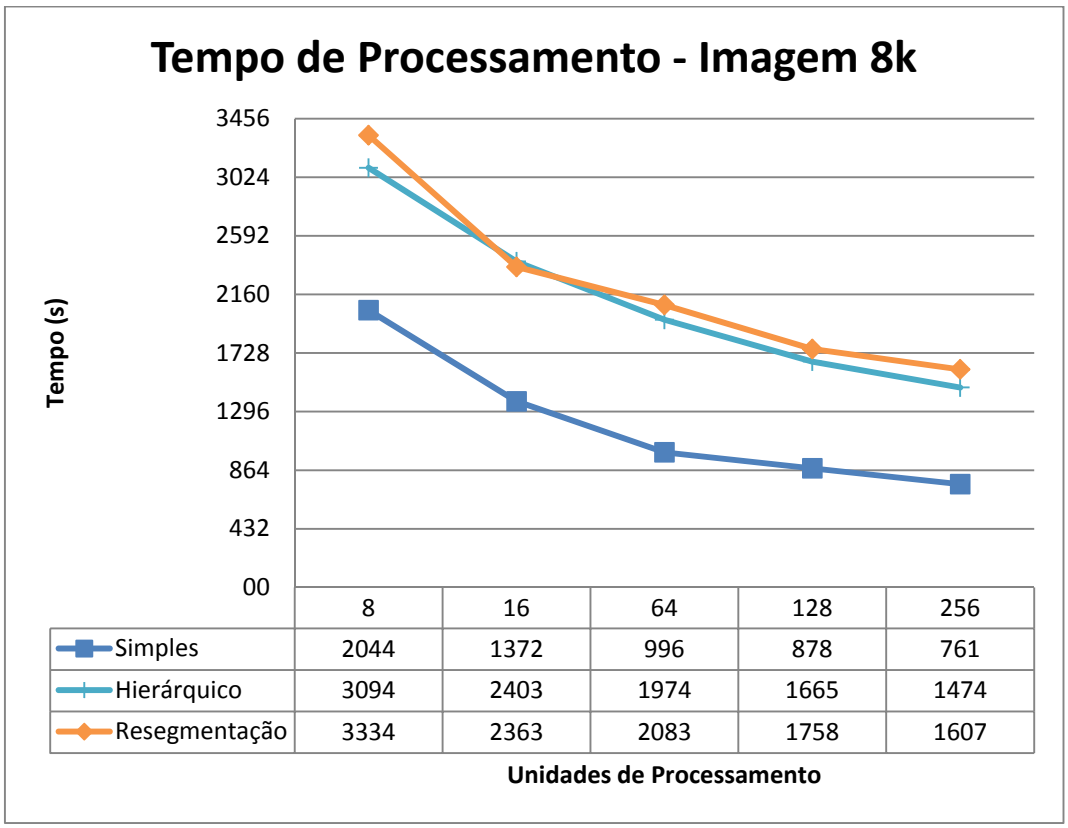


Figura 5.4. Tempo de processamento da Imagem 8k para cada método variando o número de máquinas.

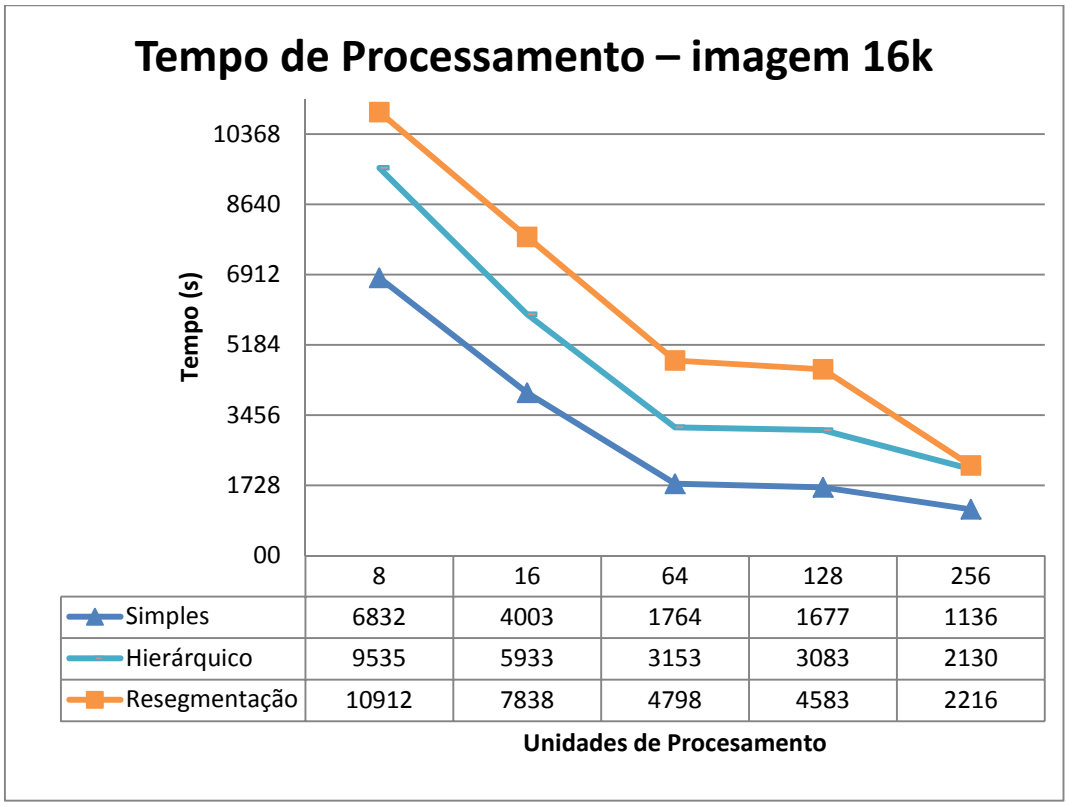


Figura 5.5. Tempo de processamento da Imagem 16k para cada método variando o número de máquinas.

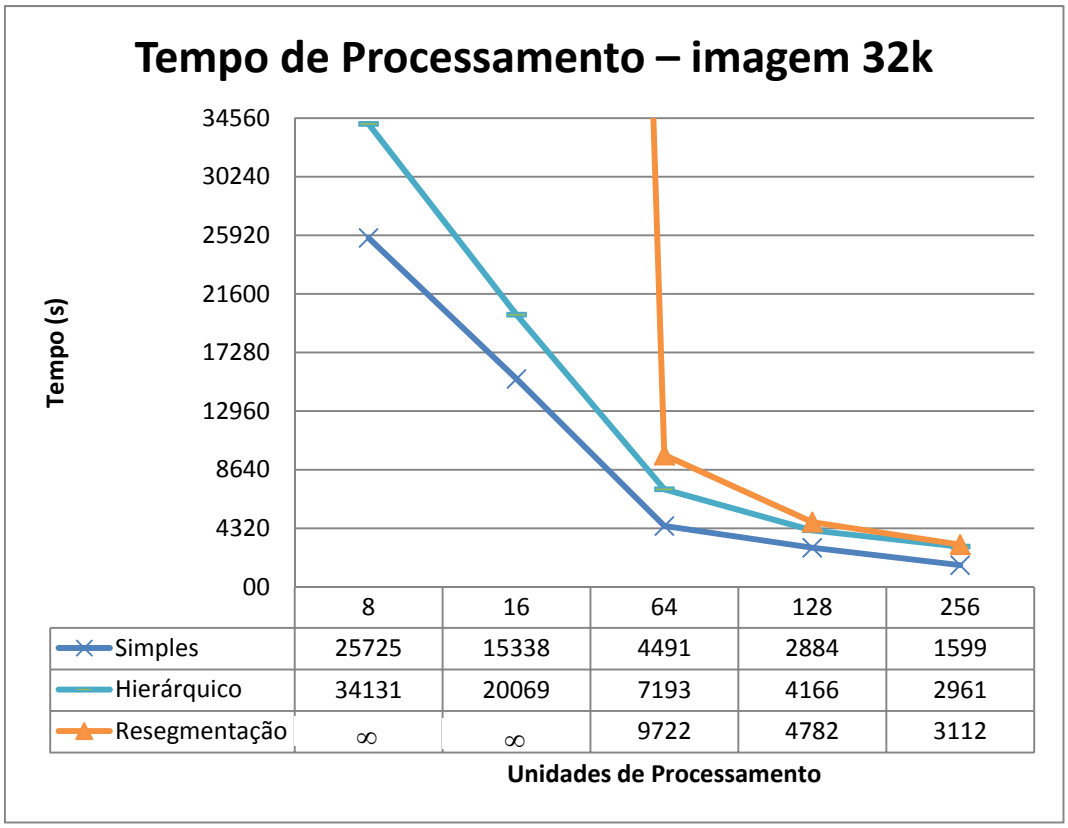
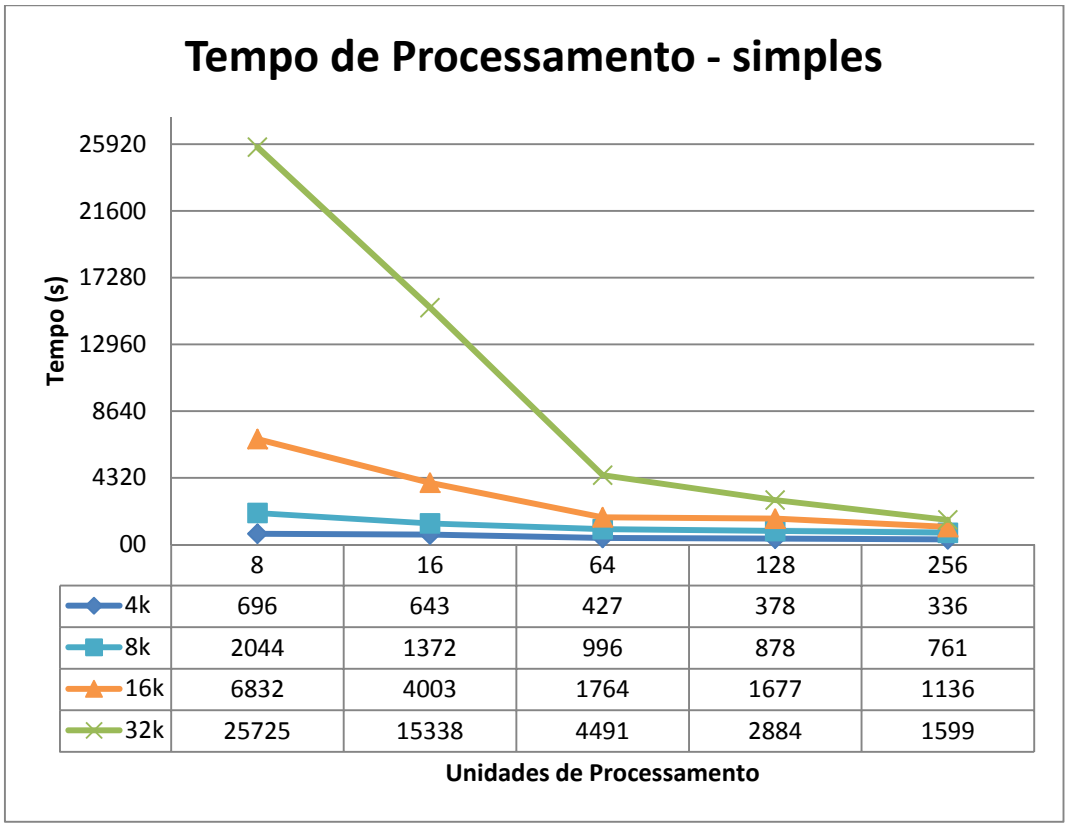
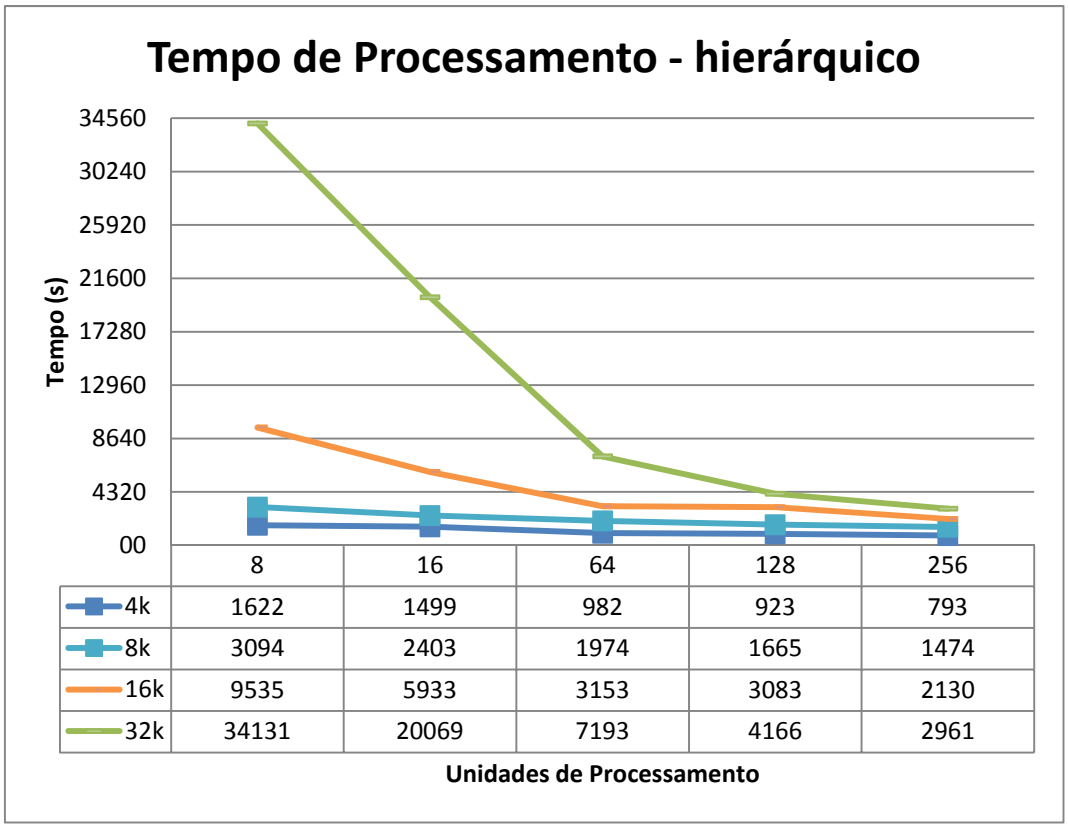


Figura 5.6. Tempo de processamento da Imagem 32k para cada método variando o número de máquinas.



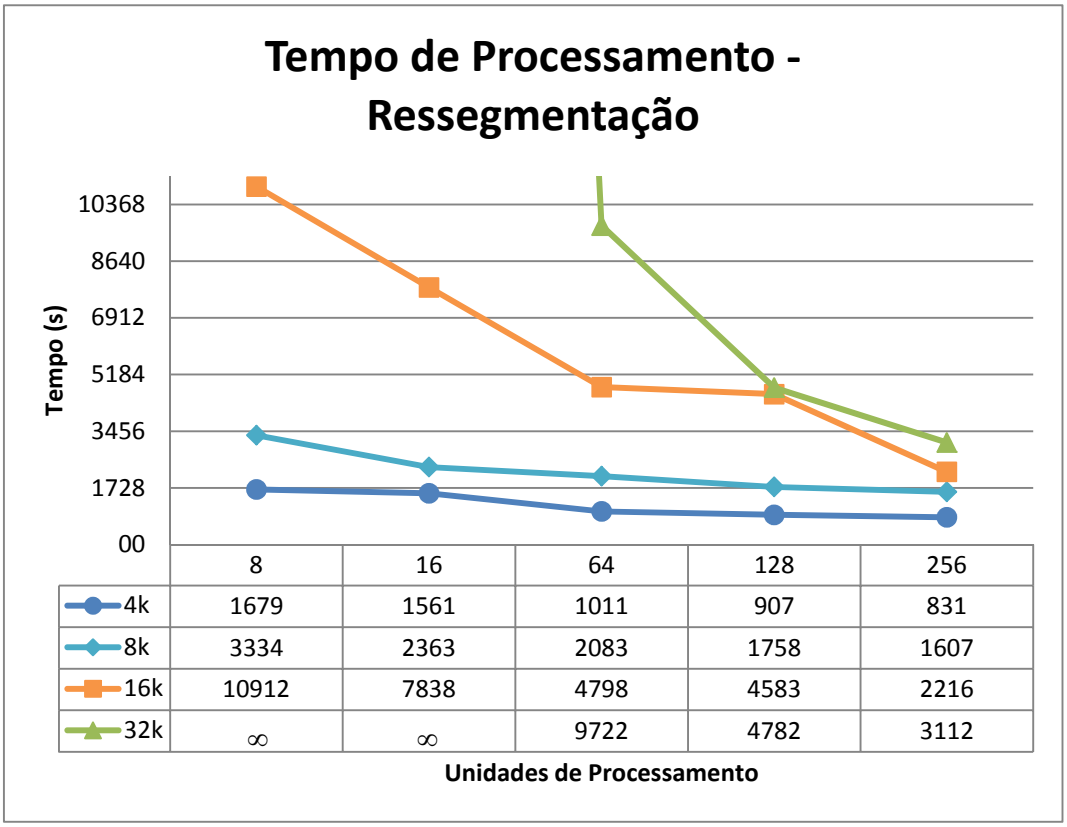
	8	16	64	128	256
4k	696	643	427	378	336
8k	2044	1372	996	878	761
16k	6832	4003	1764	1677	1136
32k	25725	15338	4491	2884	1599

Figura 5.7. Tempo de processamento para Pós-processamento Simples variando o tamanho da imagem e o número de máquinas.



	8	16	64	128	256
4k	1622	1499	982	923	793
8k	3094	2403	1974	1665	1474
16k	9535	5933	3153	3083	2130
32k	34131	20069	7193	4166	2961

Figura 5.8. Tempo de processamento para Pós-processamento Hierárquico variando o tamanho da imagem e o número de máquinas.



	8	16	64	128	256
4k	1679	1561	1011	907	831
8k	3334	2363	2083	1758	1607
16k	10912	7838	4798	4583	2216
32k	∞	∞	9722	4782	3112

Figura 5.9. Tempo de processamento para Pós-processamento Hierárquico com Ressegmentação variando o tamanho da imagem e o número de máquinas.

Para melhor verificação do ganho de desempenho de acordo com o acréscimo de máquinas, as Figuras 5.10, 5.11, 5.12 e 5.13 apresentam o *speedup* alcançado para as imagens 4k, 8k, 16k e 32k em relação à utilização de apenas uma máquina (8 unidades de processamento). Da mesma forma, as Figuras 5.14, 5.15 e 5.16 ilustram este mesmo resultado colocando em evidência os métodos de pós-processamento simples, hierárquico e hierárquico com ressegmentação.

Por intermédio dos gráficos é possível observar que o *speedup* aumenta de acordo com a quantidade de unidades de processamento disponíveis, demonstrando que a inclusão de novas máquinas acarreta em um maior ganho de desempenho. Foram obtidos *speedups* de 1,08; 1,49; 1,71 e 1,68 para as imagens 4k, 8k, 16k e 32k, respectivamente, ao se utilizar 2 máquinas, enquanto o *speedup* foi de 2,07; 2,69; 6,01 e 16,09 ao se fazer uso de 32 máquinas.

Isto também demonstra que o *speedup* aumenta conforme o tamanho da imagem cresce, pois a maior quantidade de dados torna possível explorar de forma mais eficiente o processamento distribuído. Quando o volume de dados é relativamente menor existe uma subutilização dos recursos existentes, fazendo com que o aumento de poder computacional resulte em um baixo ganho de

desempenho. As Figuras 5.14, 5.15 e 5.17 ilustram bem este fato, indicando que as imagens 4k e 8k, possuem uma curva de crescimento muito abaixo das imagens 16k e 32k.

É importante ressaltar que o processamento é limitado pela comunicação de rede existente, principalmente entre os *jobs* e que, quanto maior a quantidade de máquinas, maior é este impacto no *speedup*. Vale também informar que a quantidade de unidades de processamento indicada contempla a utilização de núcleos lógicos, os quais não possuem o mesmo poder de processamento do que os núcleos físicos.

Em relação aos três métodos, o pós-processamento simples é o que alcança os maiores valores de *speedup*. Este fato condiz com a manutenção do nível de distribuição da segmentação independente. No caso do pós-processamento hierárquico e hierárquico com ressegmentação o potencial paralelo é reduzido conforme se avança pelos diferentes níveis, diminuindo o balanceamento de carga e provocando a inutilização de unidades de processamento.

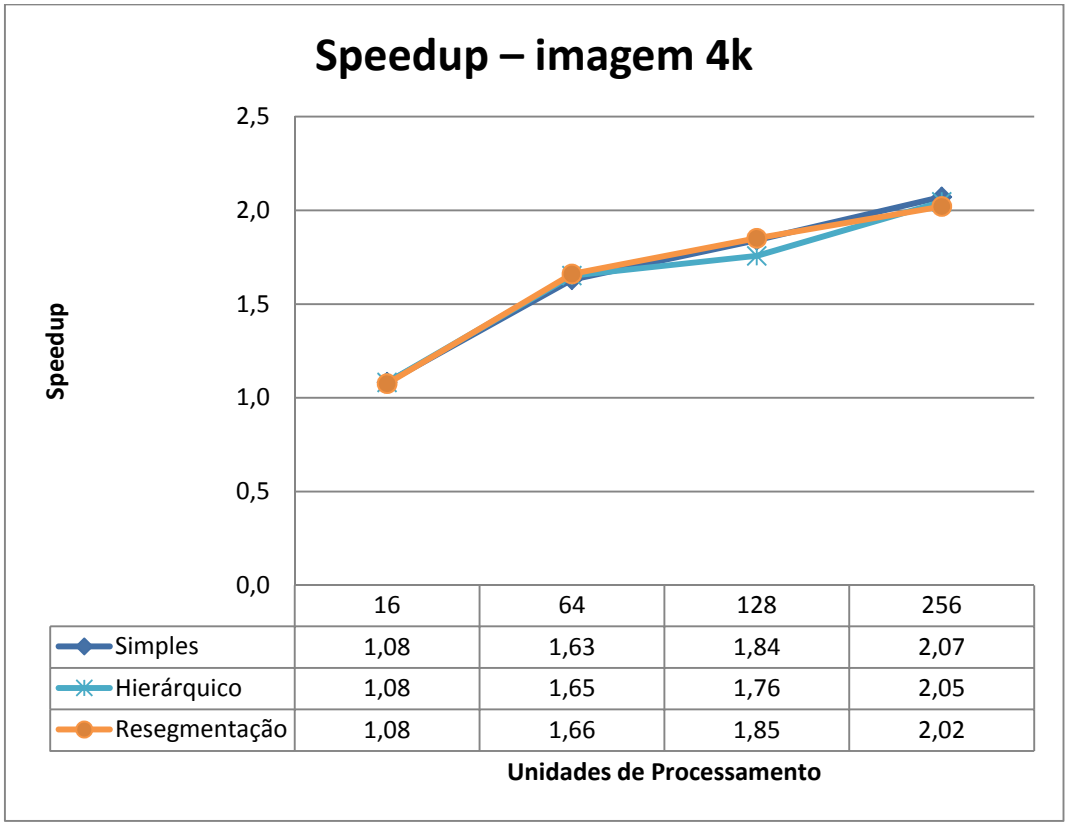


Figura 5.10. *Speedup* obtido para Imagem 4k em comparação à execução com uma única máquina (8 unidades de processamento).

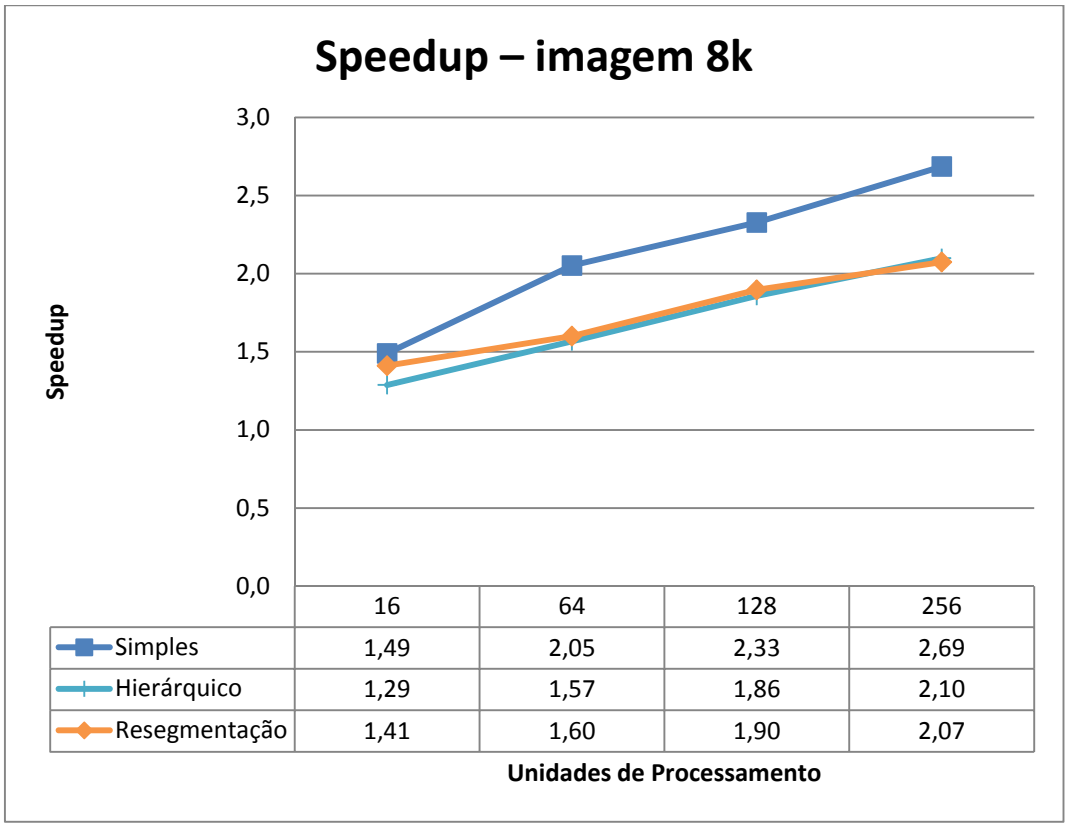


Figura 5.11. *Speedup* obtido para Imagem 8k em comparação à execução com uma única máquina (8 unidades de processamento).

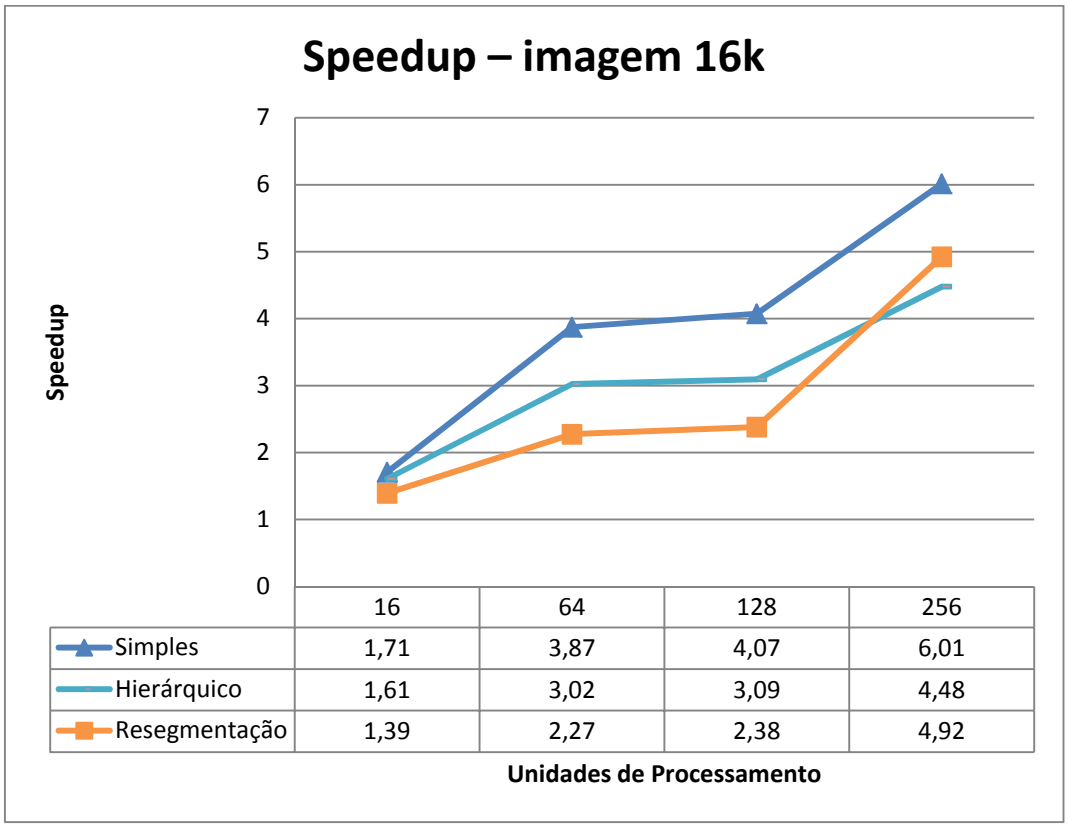


Figura 5.12. *Speedup* obtido para Imagem 16k em comparação à execução com uma única máquina (8 unidades de processamento).

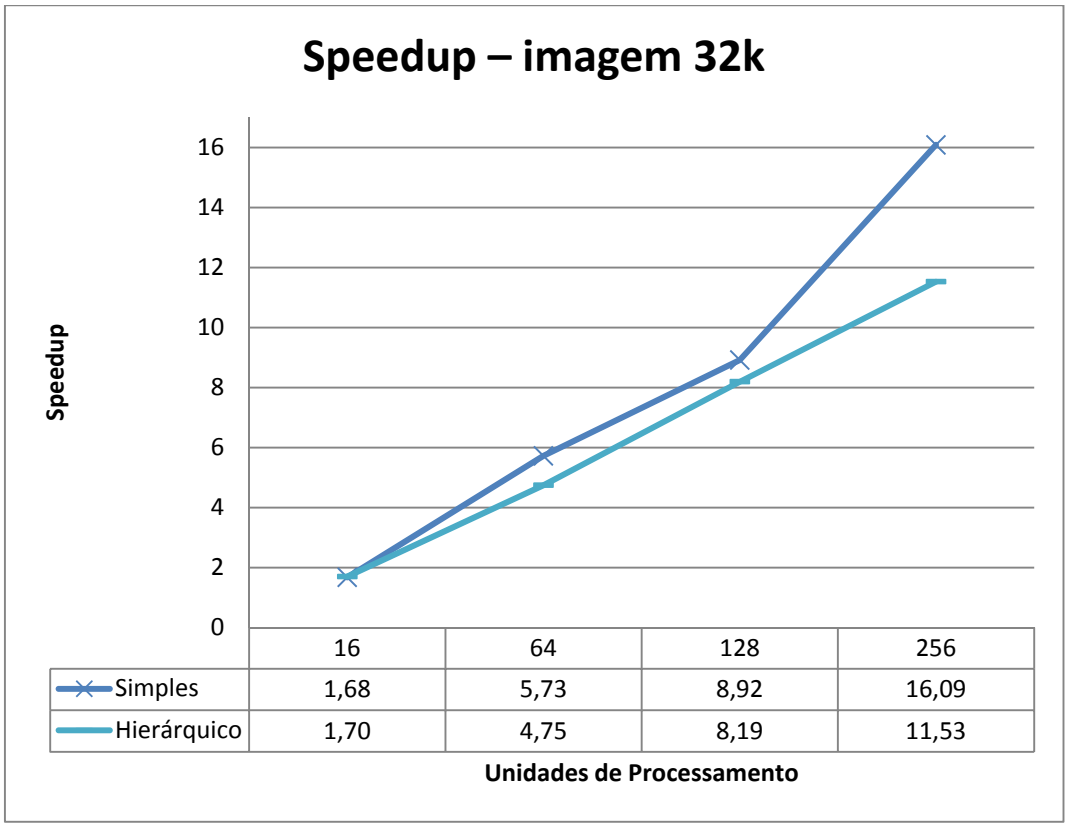


Figura 5.13. *Speedup* obtido para Imagem 32k em comparação à execução com uma única máquina (8 unidades de processamento).

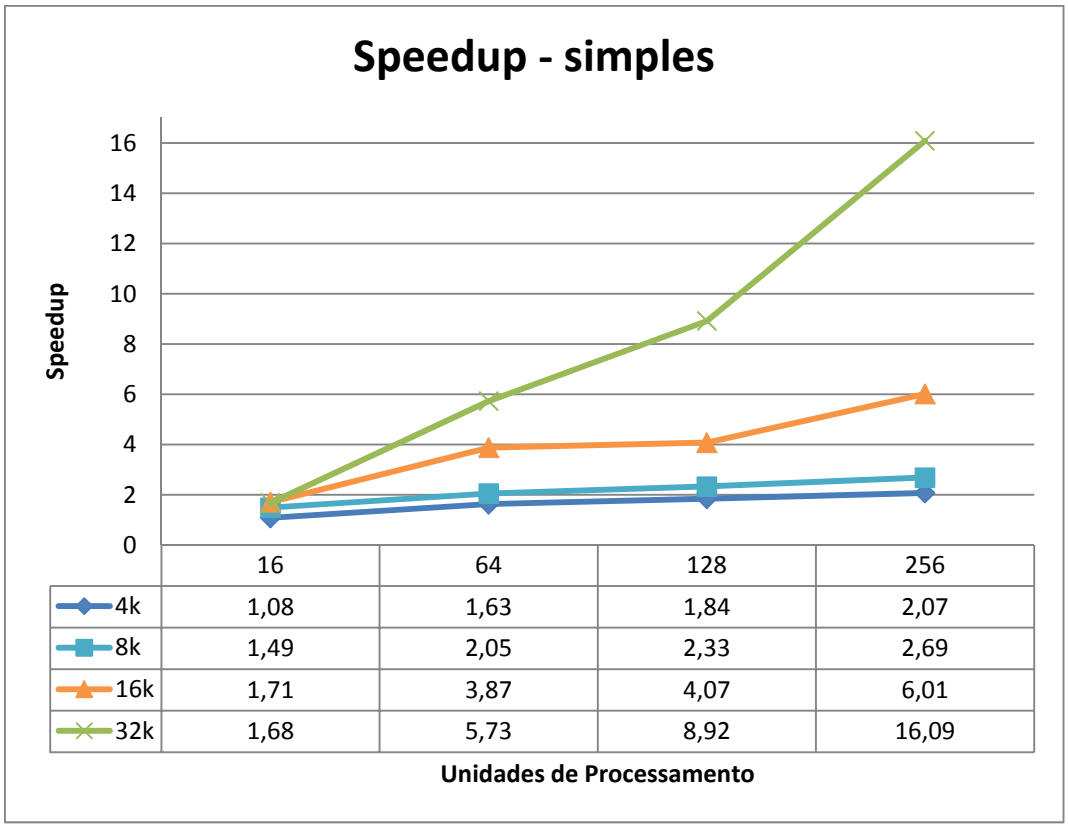
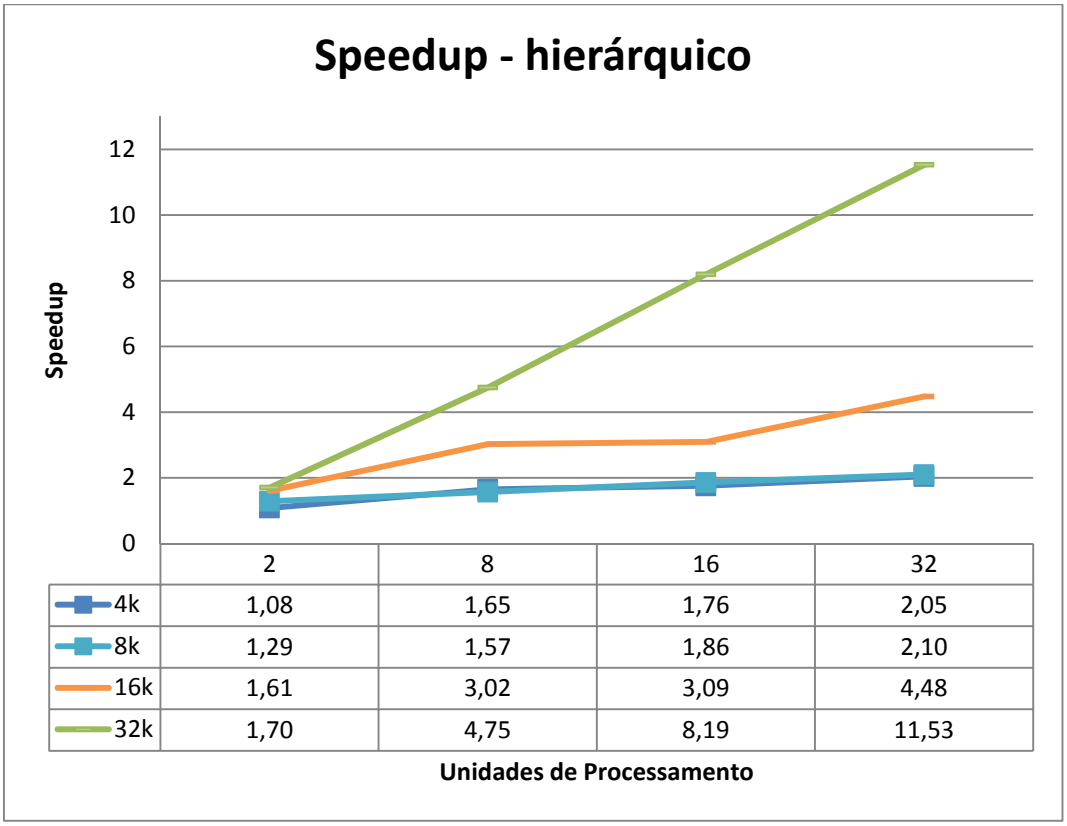
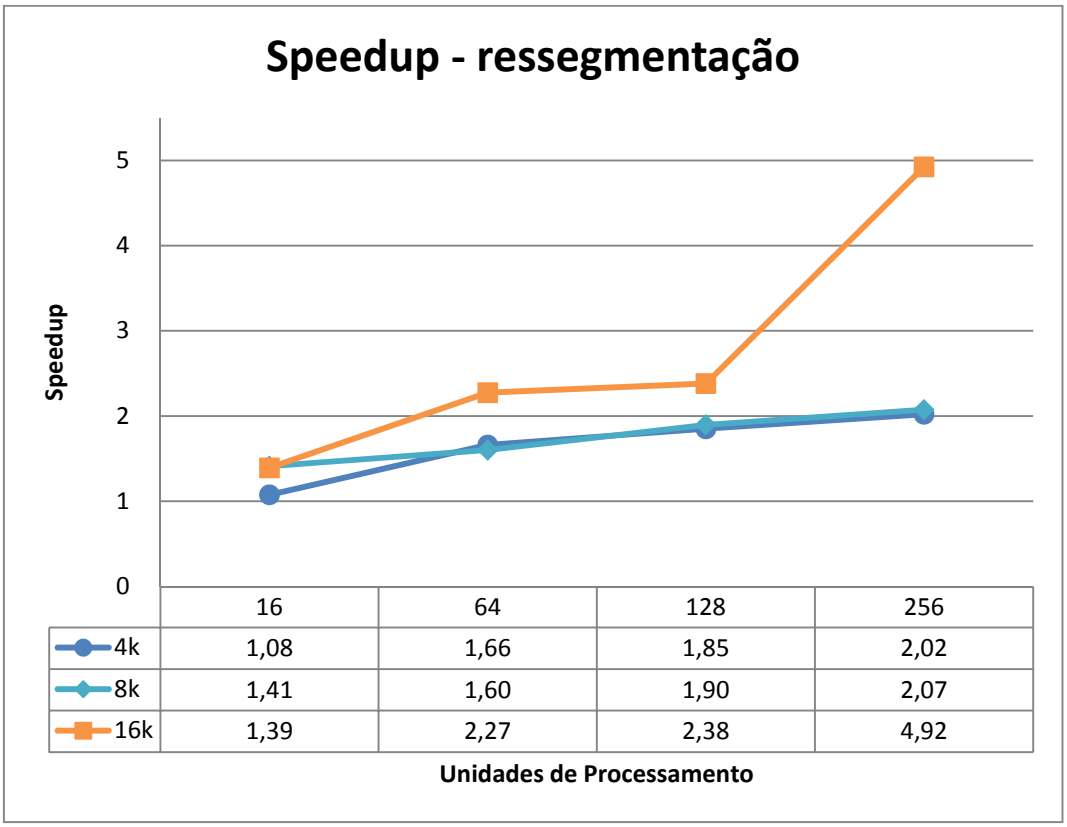


Figura 5.14. *Speedup* obtido para Pós-Processamento Simples em comparação à execução com uma única máquina (8 unidades de processamento).



	2	8	16	32
4k	1,08	1,65	1,76	2,05
8k	1,29	1,57	1,86	2,10
16k	1,61	3,02	3,09	4,48
32k	1,70	4,75	8,19	11,53

Figura 5.15. *Speedup* obtido para Pós-Processamento Hierárquico em comparação à execução com uma única máquina (8 unidades de processamento).



	16	64	128	256
4k	1,08	1,66	1,85	2,02
8k	1,41	1,60	1,90	2,07
16k	1,39	2,27	2,38	4,92

Figura 5.16. *Speedup* obtido para Pós-Processamento Hierárquico com Ressegmentação em comparação à execução com uma única máquina (8 unidades de processamento).

Outra análise interessante está relacionada com a eficiência computacional, conforme exibido nas Figuras 5.17, 5.18, 5.19 e 5.20. As figuras demonstram a eficiência do processamento de cada imagem em relação à quantidade de máquinas (unidades de processamento) utilizadas, tendo como base a execução em uma única máquina (8 unidades de processamento). Novamente, uma visão alternativa dos gráficos em relação a cada método de pós-processamento é exibida nas Figuras 5.21, 5.22 e 5.23.

Os resultados mostram que para o mesmo número de máquinas e um determinado método, a eficiência aumenta de acordo com o aumento do tamanho da imagem. Isso significa que nos experimentos realizados a capacidade de processamento não atingiu seu máximo e uma imagem maior tende a resultar em um maior ganho computacional. A exceção, conforme visto na Fig. 21, é a imagem 32k processada em apenas 2 máquinas pelo pós-processamento simples, que obtém uma eficiência inferior a obtida pela imagem 16k. Assim, para uma dada quantidade de máquinas, existe um tamanho de imagem a partir do qual a eficiência deixa de crescer.

Os resultados também demonstram que a eficiência diminui à medida que o número de máquinas aumenta. Este fato está diretamente relacionado com as perdas provenientes da comunicação, que aumentam proporcionalmente com o número de máquinas. Além disso, outro fator a ser considerado é a questão de exploração máxima da capacidade computacional existente. Para um menor número de máquinas, o ponto máximo de eficiência é encontrado antes e, assim, conforme o tamanho da imagem cresce, a eficiência ao se utilizar uma maior quantidade de máquinas também cresce. O ponto de máximo para um maior número de máquinas estará, portanto, deslocado mais para direita, ou seja, será encontrado para tamanhos maiores de imagem.

Por fim, também é possível afirmar que o método de pós-processamento simples é o que apresenta maior eficiência quando comparado aos outros. A explicação é análoga àquela apresentada anteriormente para justificar o seu maior ganho de desempenho computacional.

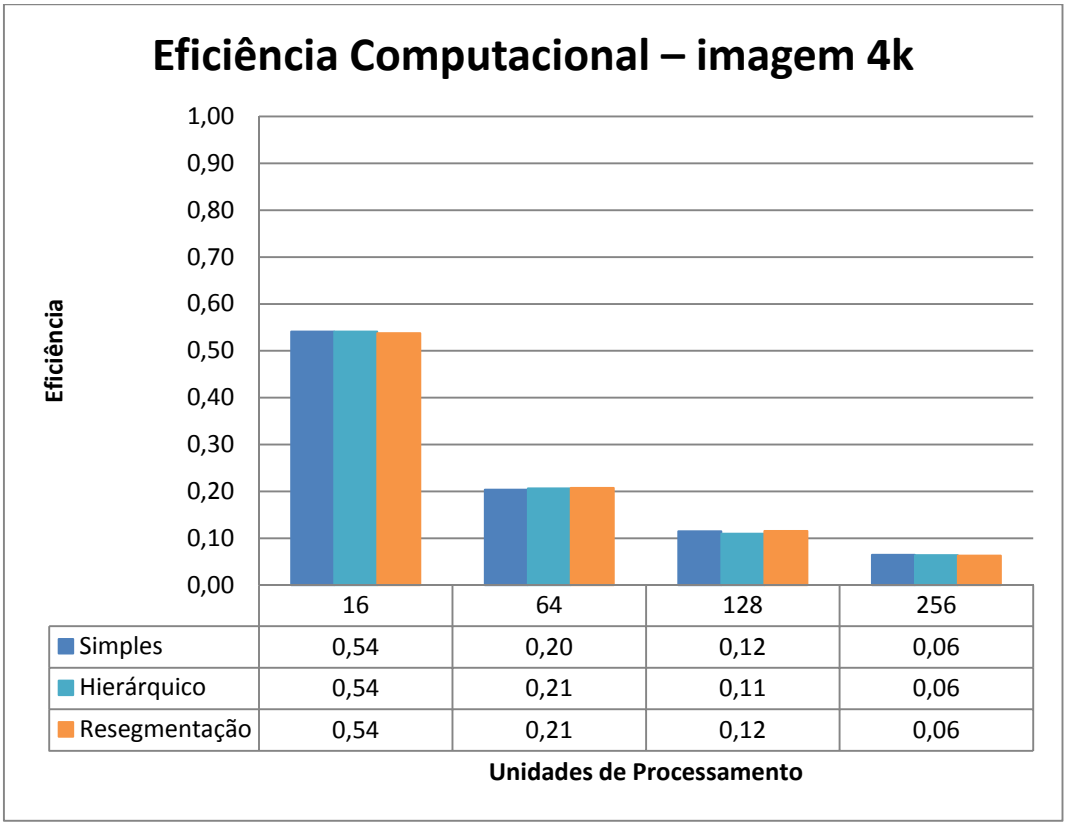


Figura 5.17. Eficiência computacional resultante para Imagem 4k tendo como base uma única máquina (8 unidades de processamento).

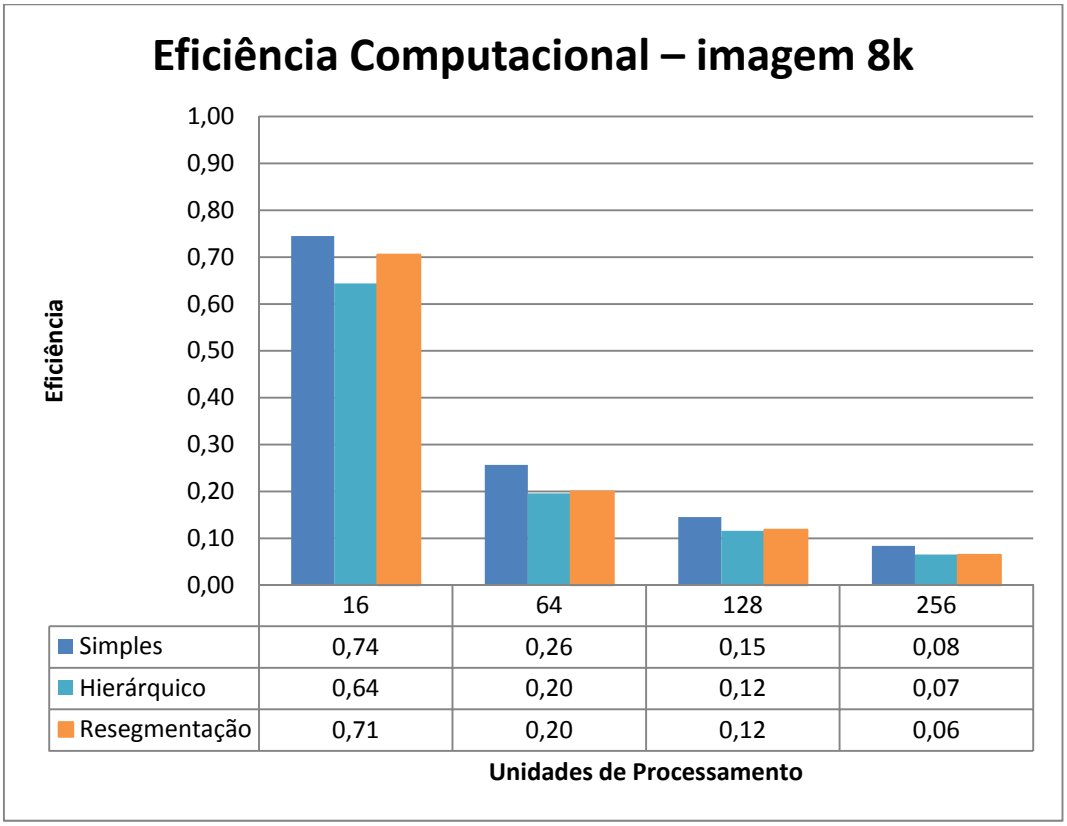


Figura 5.18. Eficiência computacional resultante para Imagem 8k tendo como base uma única máquina (8 unidades de processamento).

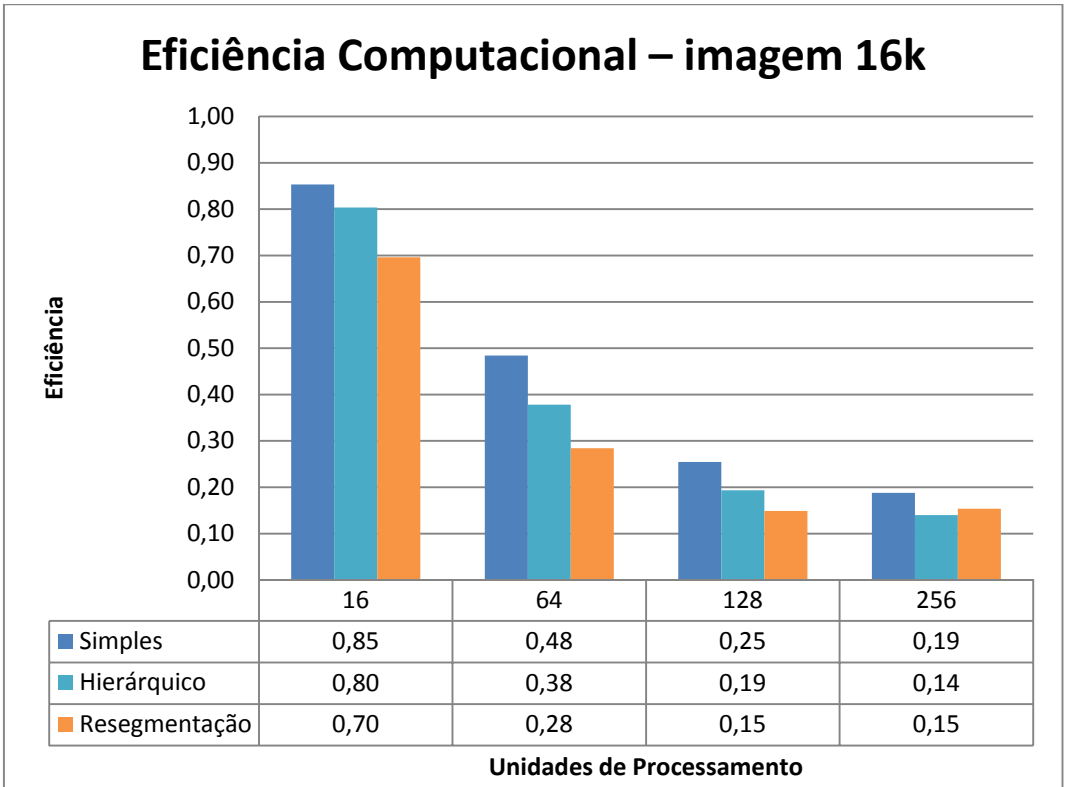


Figura 5.19. Eficiência computacional resultante para Imagem 16k tendo como base uma única máquina (8 unidades de processamento).

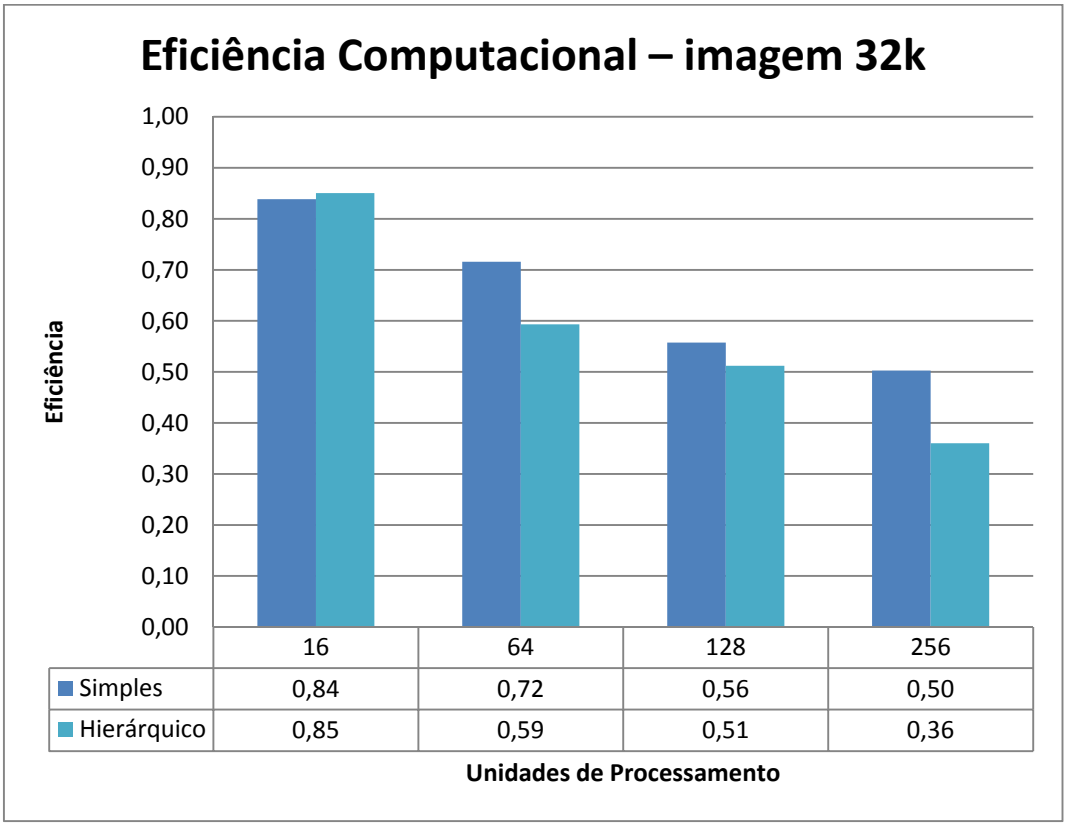
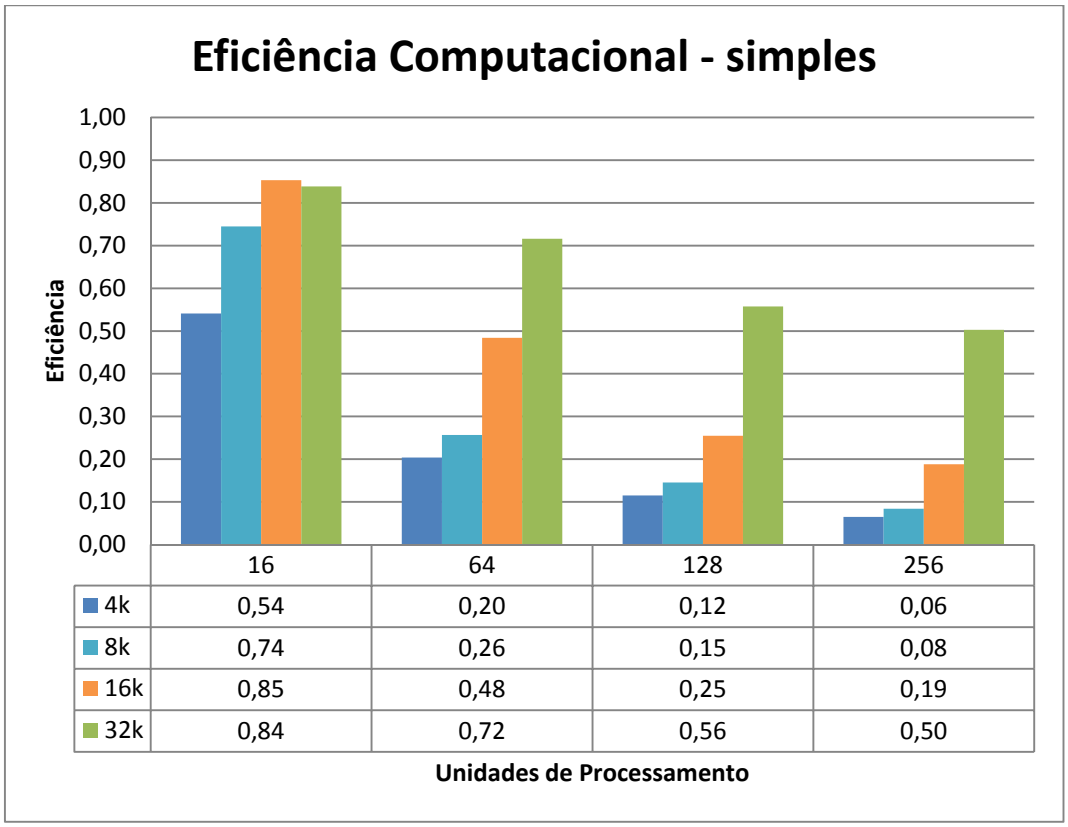
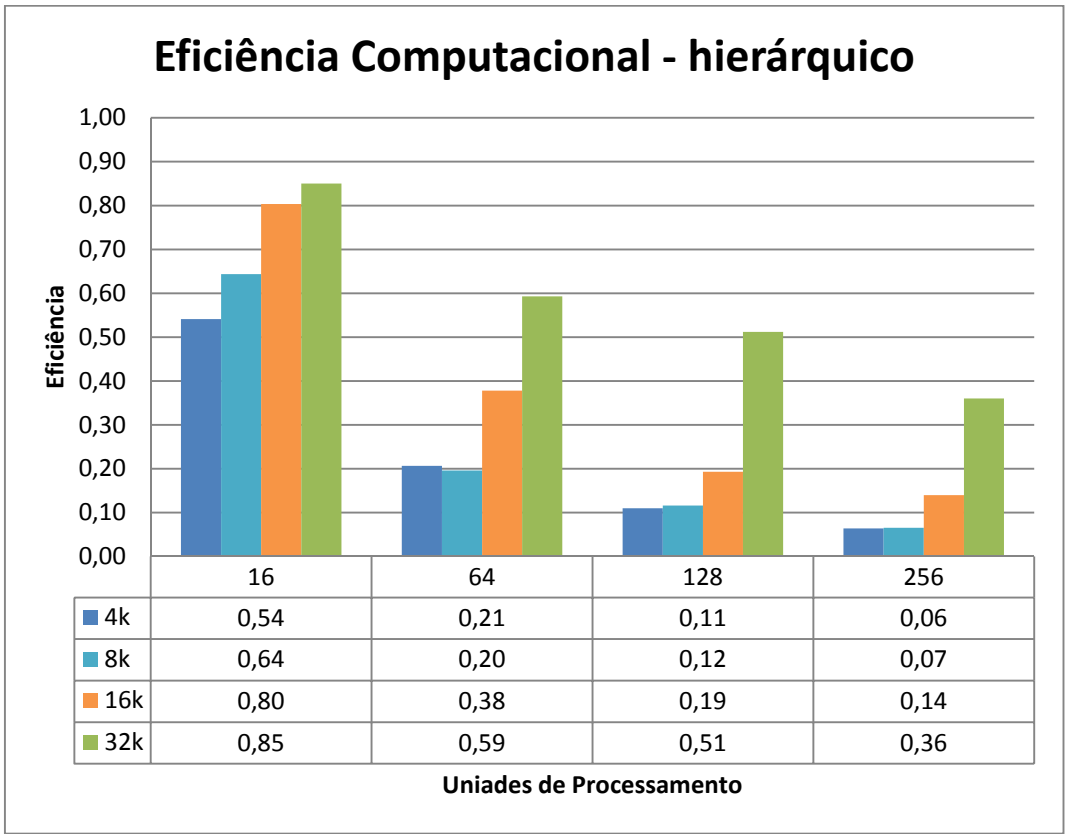


Figura 5.20. Eficiência computacional resultante para Imagem 32k tendo como base uma única máquina (8 unidades de processamento).



	16	64	128	256
4k	0,54	0,20	0,12	0,06
8k	0,74	0,26	0,15	0,08
16k	0,85	0,48	0,25	0,19
32k	0,84	0,72	0,56	0,50

Figura 5.21. Eficiência computacional resultante para Pós-Processamento Simples tendo como base uma única máquina (8 unidades de processamento).



	16	64	128	256
4k	0,54	0,21	0,11	0,06
8k	0,64	0,20	0,12	0,07
16k	0,80	0,38	0,19	0,14
32k	0,85	0,59	0,51	0,36

Figura 5.22. Eficiência computacional resultante para Pós-Processamento Hierárquico tendo como base uma única máquina (8 unidades de processamento).

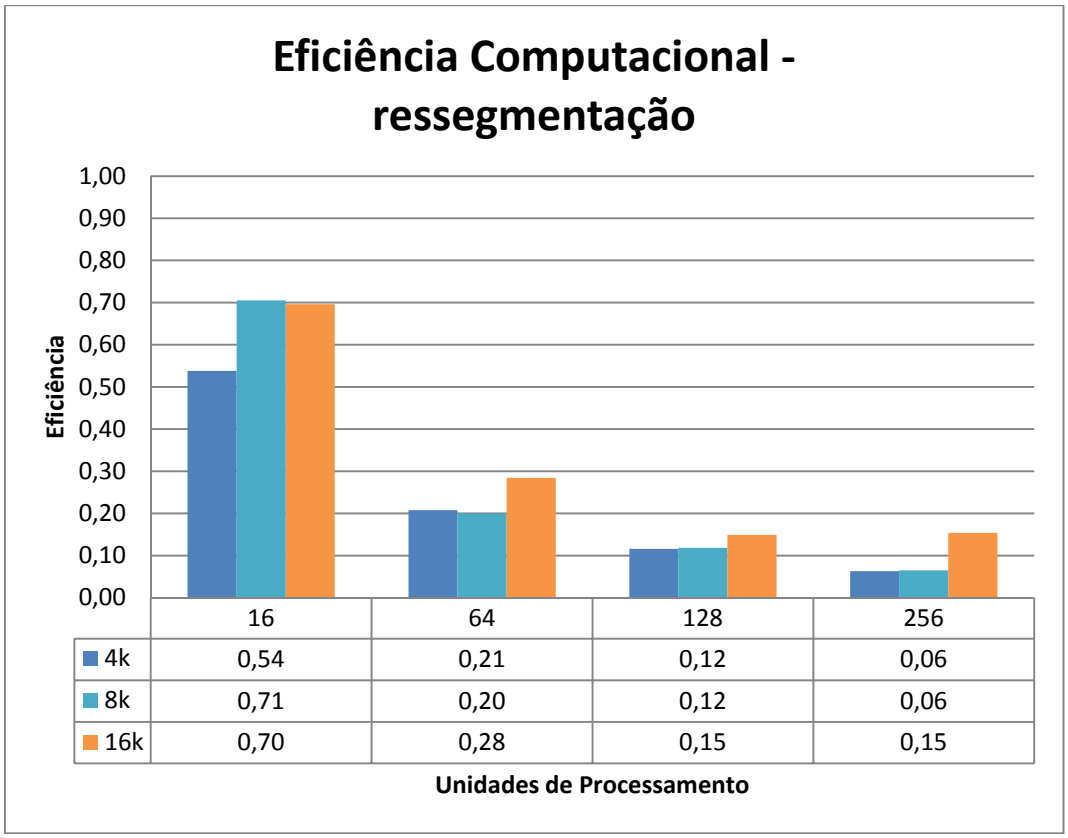


Figura 5.23. Eficiência computacional resultante para Pós-Processamento Hierárquico com Ressegmentação tendo como base uma única máquina (8 unidades de processamento).

5.3.Avaliação de qualidade

A fim de validar o correto funcionamento do método apresentado, uma avaliação de qualidade do resultado foi realizada. Para tanto, o método com as três diferentes estratégias de pós-processamento foi executado na nuvem, sendo cada resultado analisado de acordo com o delineamento dos segmentos gerados e da presença de artefatos na imagem. Foram utilizadas apenas duas instâncias para processamento a fim de reduzir o custo de execução dos experimentos.

5.3.1.Base de Dados

A imagem Quickbird anteriormente chamada de 4k (Figura 5.1) de tamanho 4000 x 4000 *pixels*, resolução de 0,61m e 4 bandas espectrais foi selecionada para os testes envolvendo a qualidade do resultado. Para verificar o funcionamento do método com imagens de diferentes sensores também foram incluídos um recorte de uma cena do satélite GeoEye-1 de tamanho 3342 x 1692 *pixels*, resolução 0,5m e 4 bandas espectrais – denominado GeoEye - (Figura 5.24); e um recorte de uma

imagem aérea de 0,1m de resolução, 4524 x 3562 *pixels* e 3 bandas espectrais – denominada Aérea - (Figura 5.25).

Células geográficas/*tiles* de tamanho 1024 x 1024 foram definidas para realizar a divisão da imagem, conforme discutido na seção 5.2.2. Os **p**arâmetros de segmentação podem ser observados na Tabela 5.2.

Figura 5.24. Imagem de teste GeoEye.

Figura 5.25. Imagem de teste Aérea.

Imagem	Escala	Cor / forma	Compacidade / Suavidade	Peso das Bandas
4k	41	0,84	0,8	1,1,1,1
GeoEye	30	0,75	0,8	1,1,1,1
Aérea	42	0,7	0,8	1,1,1

Tabela 5.2: Conjunto de parâmetros utilizado para segmentação.

5.3.2.Resultados e Discussões

As Figuras 5.26, 5.27 e 5.28 apresentam, respectivamente, uma parte do resultado de segmentação distribuída da imagem 4k para os métodos contendo o pós-processamento simples, hierárquico e hierárquico com ressegmentação. As figuras encontram-se rotacionadas para melhor visualização. Conforme esperado, o pós-processamento simples (Figura 5.26) é o que apresenta a maior quantidade de artefatos. É possível observar algumas linhas referente às bordas dos *tiles* em determinados locais da imagem. Marcações em vermelho indicam a direção de algumas destas.

Ao comparar a Figura 5.27 com a Figura 5.26 nota-se uma elevada redução de artefatos na imagem. Ainda assim é possível verificar a presença de algumas irregularidades nas bordas dos *tiles.* Este problema é porque os segmentos que já cresceram não podem alterar suas decisões de fusão anteriores. Assim, a única possibilidade é a fusão de segmentos já existentes. Já a Figura 5.28 não apresenta artefatos indesejáveis nas bordas dos *tiles*, demostrando que o pós-processamento hierárquico com ressegmentação elimina este tipo de problema. A ideia é prover liberdade para os segmentos crescerem novamente desde seu início.

A Figura 5.29 apresenta uma comparação do resultado de cada uma das estratégias e exibi-os lado a lado de forma a facilitar a visualização das diferenças. Na Figura 5.29(a), é possível observar a presença de artefatos na direção das duas linhas horizontais marcadas em vermelho nas laterais da figura. Seguindo estas linhas, verifica-se na Figura 5.29 (b) a quantidade de artefatos é reduzida, principalmente dentro do estádio. Por fim, na Figura 5.29(c) nota-se que não existem artefatos. Outra observação importante é que a parte mais interna dos *tiles* é exatamente idêntica, pois a segmentação independente gera os mesmos segmentos, que são armazenados antes do pós-processamento ser iniciado.

Figura 5.26. Parte da segmentação utilizando pós-processamento simples para 4K. Obs.: Imagem rotacionada.

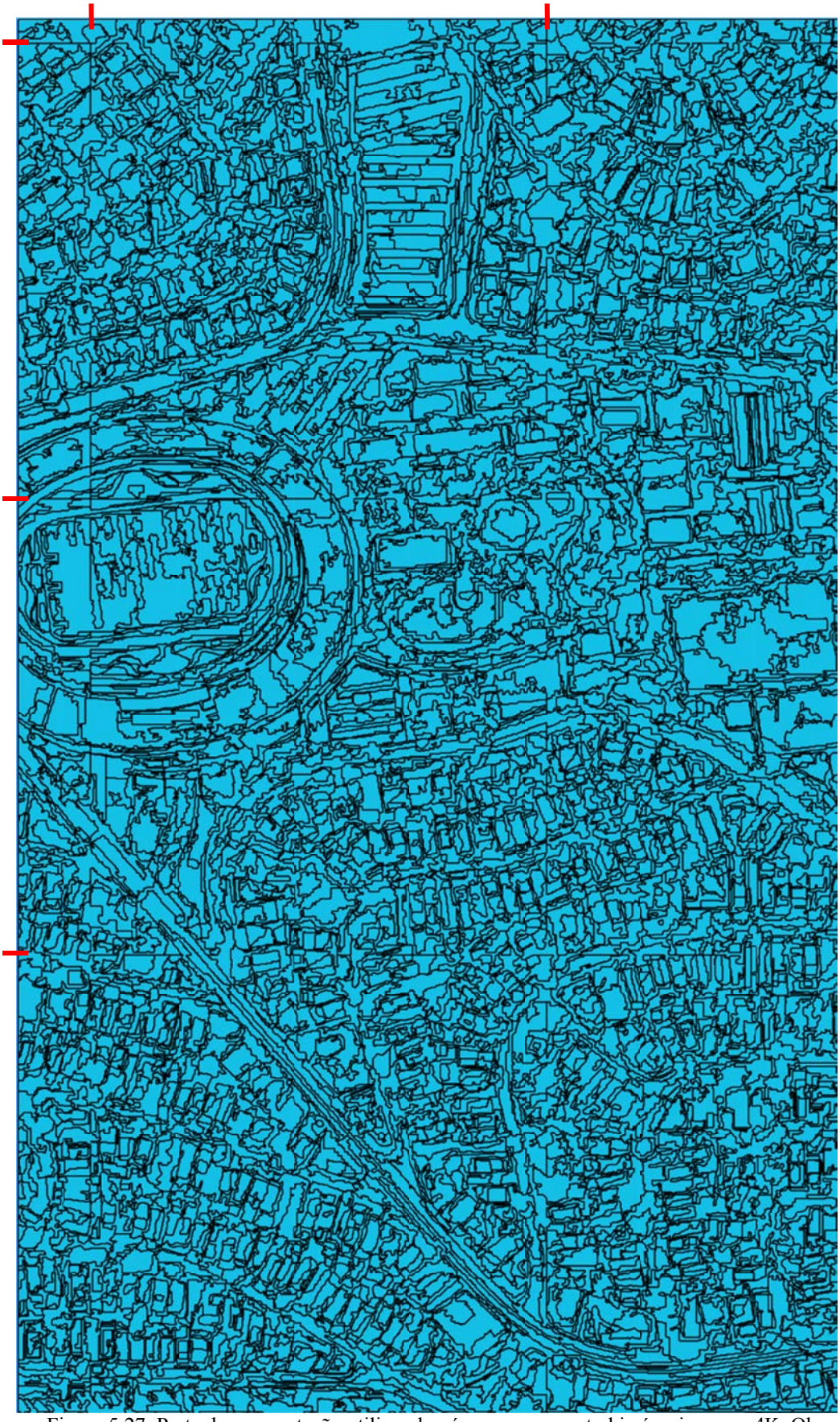


Figura 5.27. Parte da segmentação utilizando pós-processamento hierárquico para 4K. Obs.: Imagem rotacionada.

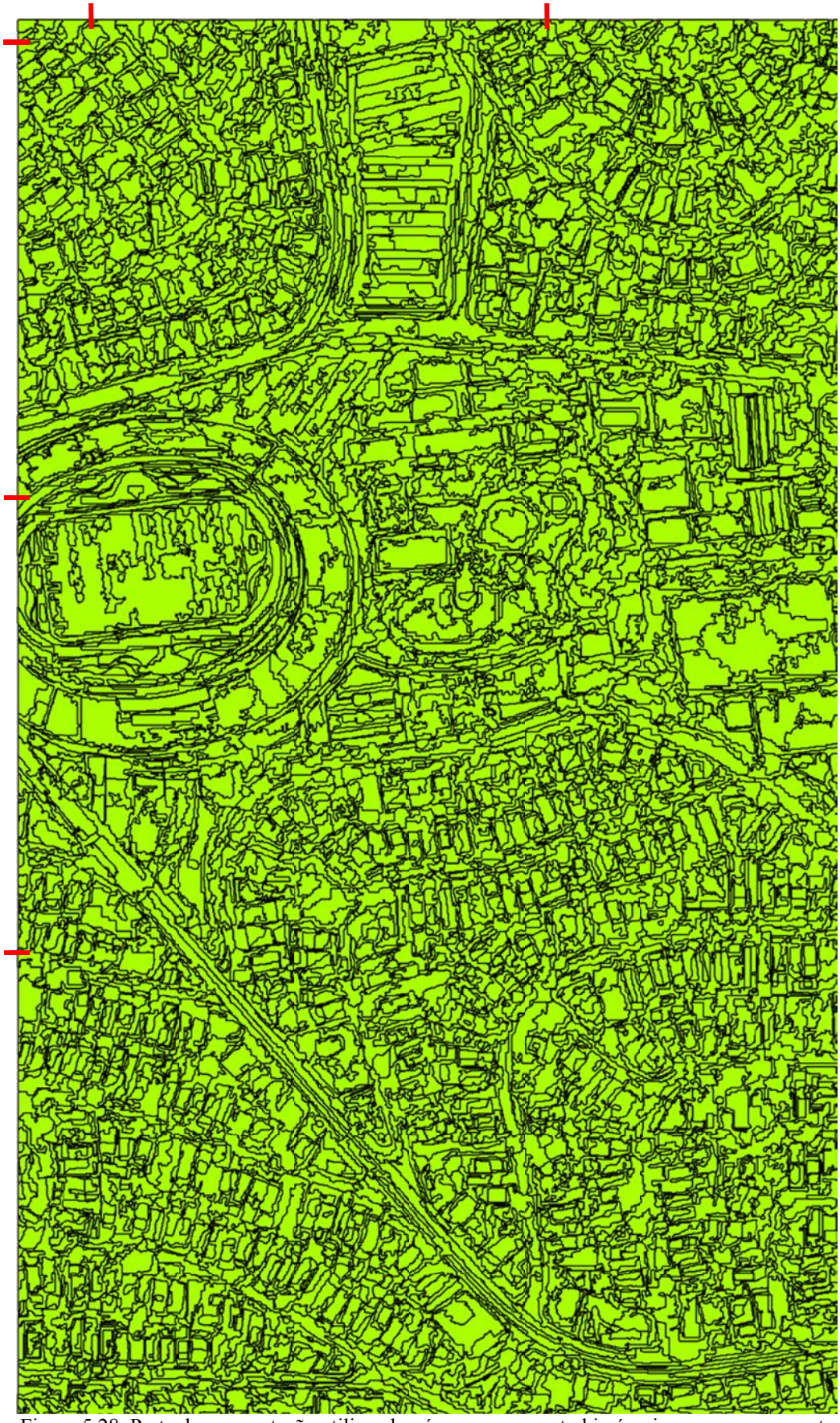


Figura 5.28. Parte da segmentação utilizando pós-processamento hierárquico com ressegmentação para 4K. Obs.: Imagem rotacionada.

(a) (b) (c)

Figura 5.29. Comparação entre as estratégias de pós-processamento: simples (a), hierárquico (b) e hierárquico com segmentação (c) para imagem 4K.

Uma análise similar pode ser também realizada para a imagem GeoEye. Os resultados dos métodos de pós-processamento simples, hierárquico e hierárquico com ressegmentação são apresentados, respectivamente, nas Figuras 5.30, 5.31 e 5.32. Neste caso, o resultado do pós-processamento simples é bastante semelhante ao resultado do pós-processamento hierárquico e ambos possuem artefatos sobre linhas horizontais e verticais que correspondem às fronteiras dos *tiles* (indicadas em vermelho). A estratégia com ressegmentação, quando observada de maneira geral, não produz uma segmentação tão diferente das demais. Este fato está relacionado com a quantidade de segmentos internos produzidos pela etapa de segmentação independente. Como esperado, a disparidade está localizada próxima às bordas dos *tiles* e a principal distinção é a ausência de artefatos.

A Figura 5.33 exibe lado a lado recortes dos resultados da segmentação da imagem GeoEye para as três estratégias de pós-processamento. Nota-se que o pós-processamento simples, Figura 5.33(a), e o pós-processamento hierárquico, Figura 5.33(b), produzem praticamente o mesmo resultado. Em ambos se observa a presença de artefatos nas direções indicadas nas linhas em vermelho (direção das bordas dos *tiles*). No caso do pós-processamento hierárquico com ressegmentação, Figura 5.33(c), esses artefatos não estão presentes. Outro detalhe a ser verificado é a existência de segmentos maiores nas duas primeiras estratégias quando comparados com a última. Este fato está relacionado com a liberdade de crescimento oriunda da ressegmentação que permite que os segmentos sejam reajustados, diferente das outras duas estratégias que só é possibilitam a fusão de segmentos pré-existentes.

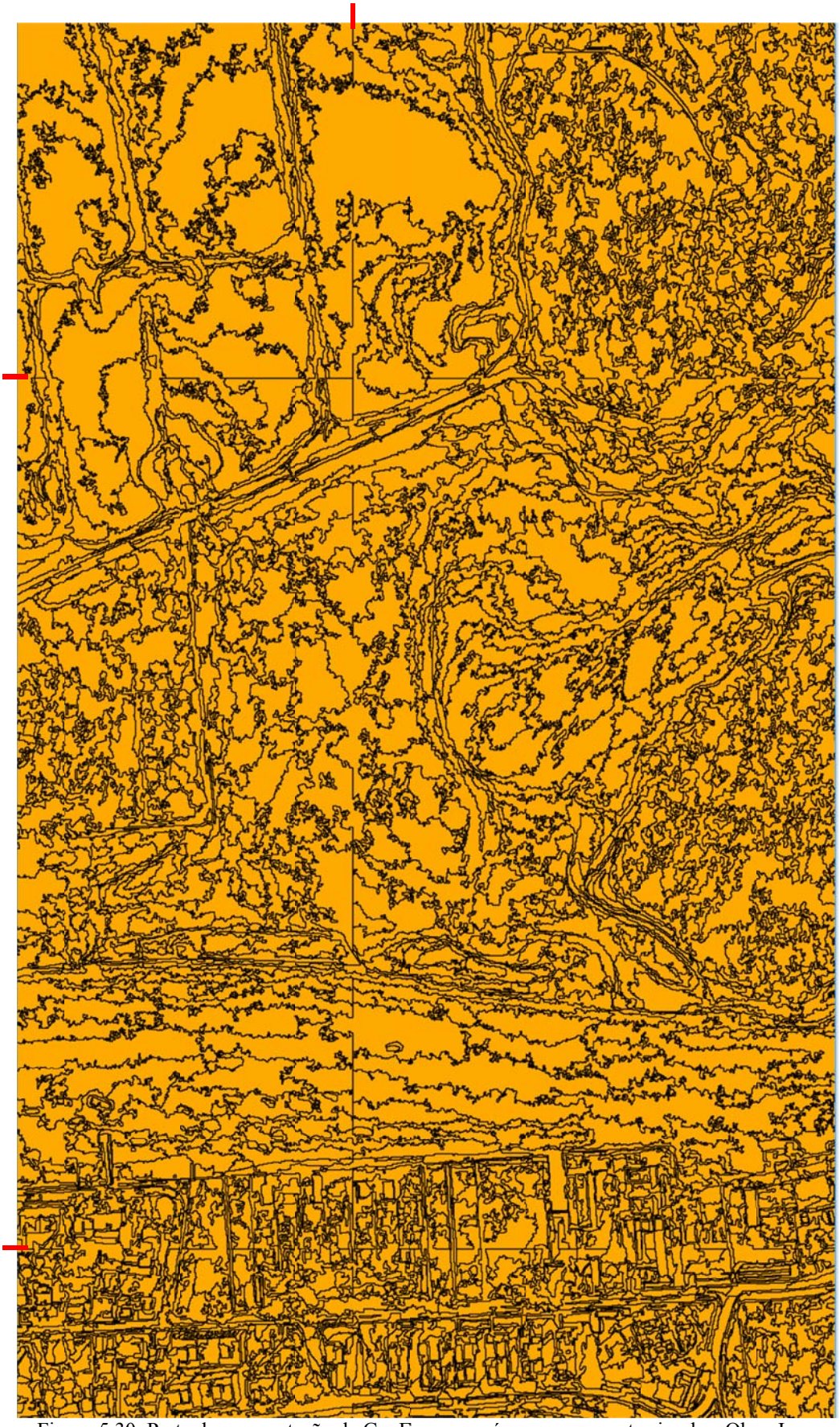


Figura 5.30. Parte da segmentação de GeoEye com pós-processamento simples. Obs.: Imagem rotacionada.

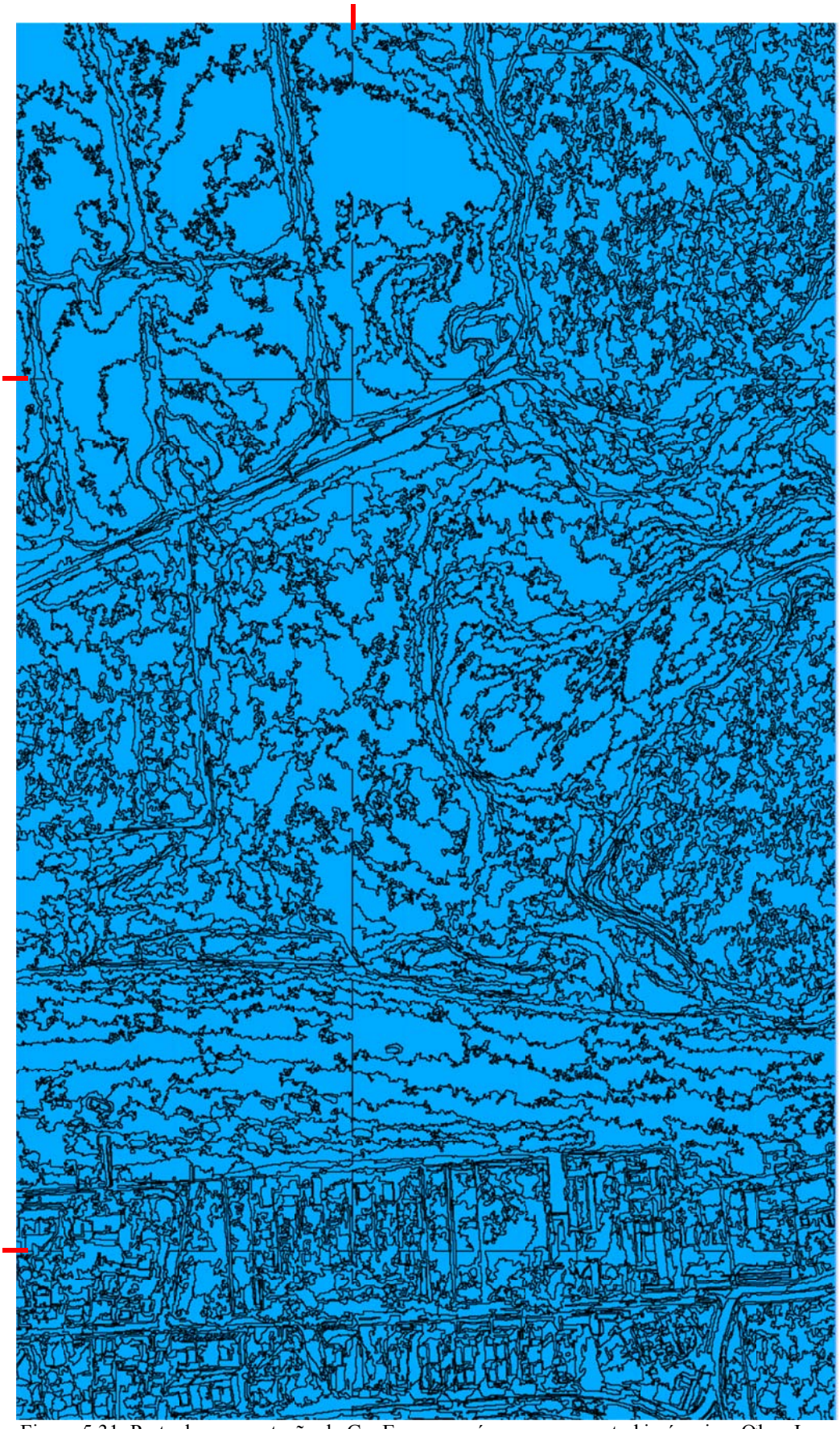


Figura 5.31. Parte da segmentação de GeoEye com pós-processamento hierárquico. Obs.: Imagem rotacionada.

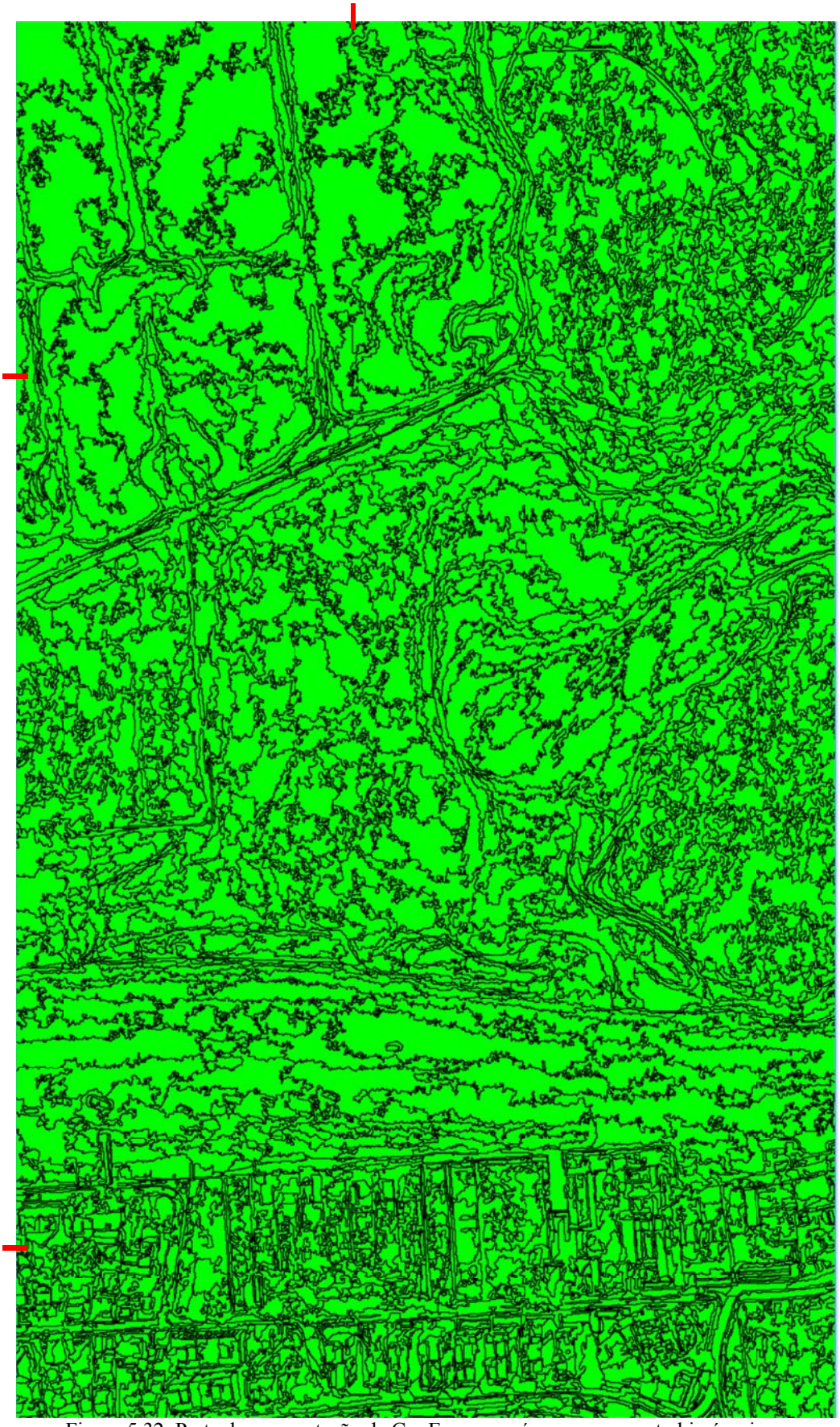
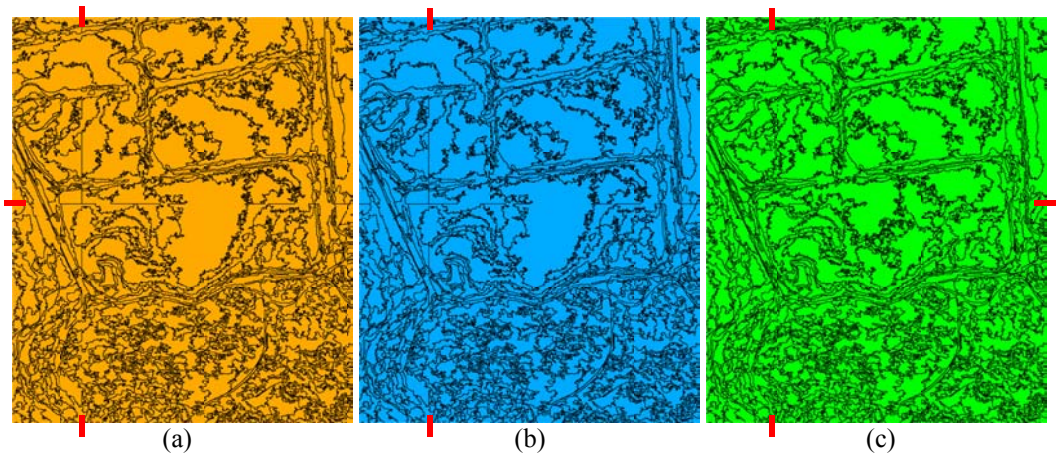


Figura 5.32. Parte da segmentação de GeoEye com pós-processamento hierárquico com ressegmentação. Obs.: Imagem rotacionada.



(a) (b) (c)

Figura 5.33. Comparação entre as estratégias de pós-processamento: simples (a), hierárquico (b) e hierárquico com segmentação (c) para imagem GeoEye.

Por fim, as Figuras 5.34, 5.35 e 5.36 ilustram os resultados das três estratégias de pós-processamento (simples, hierárquico e hierárquico com ressegmentação) referentes à imagem Aérea. As imagens encontram-se rotacionadas para melhor visualização. A análise destes resultados conduz a observações e conclusões comparáveis àquelas realizadas com as outras duas imagens de teste.

Para finalizar, a Figura 5.37 apresenta recortes da segmentação da imagem Aérea gerados pelas três estratégias. Novamente é possível verificar a presença dos artefatos na direção das linhas vermelhas destacadas para as estratégias de pós-processamento simples, Figura 5.37(a), e hierárquico, Figura 5.37(b), enquanto uma segmentação mais consistente, sem a presença de artefatos, é observada no resultado do pós-processamento hierárquico com ressegmentação Figura 5.37(c).

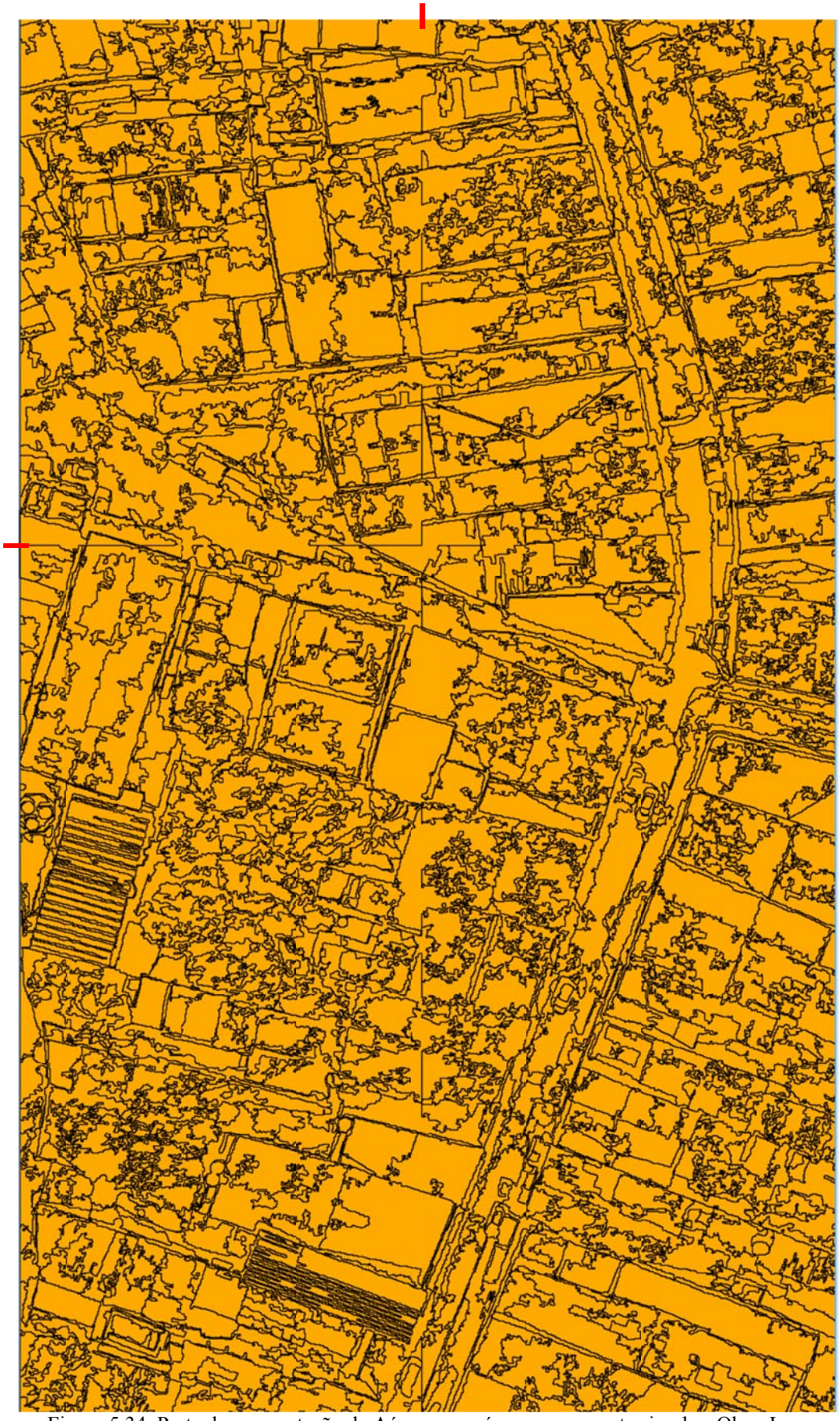


Figura 5.34. Parte da segmentação de Aérea com pós-processamento simples. Obs.: Imagem rotacionada.

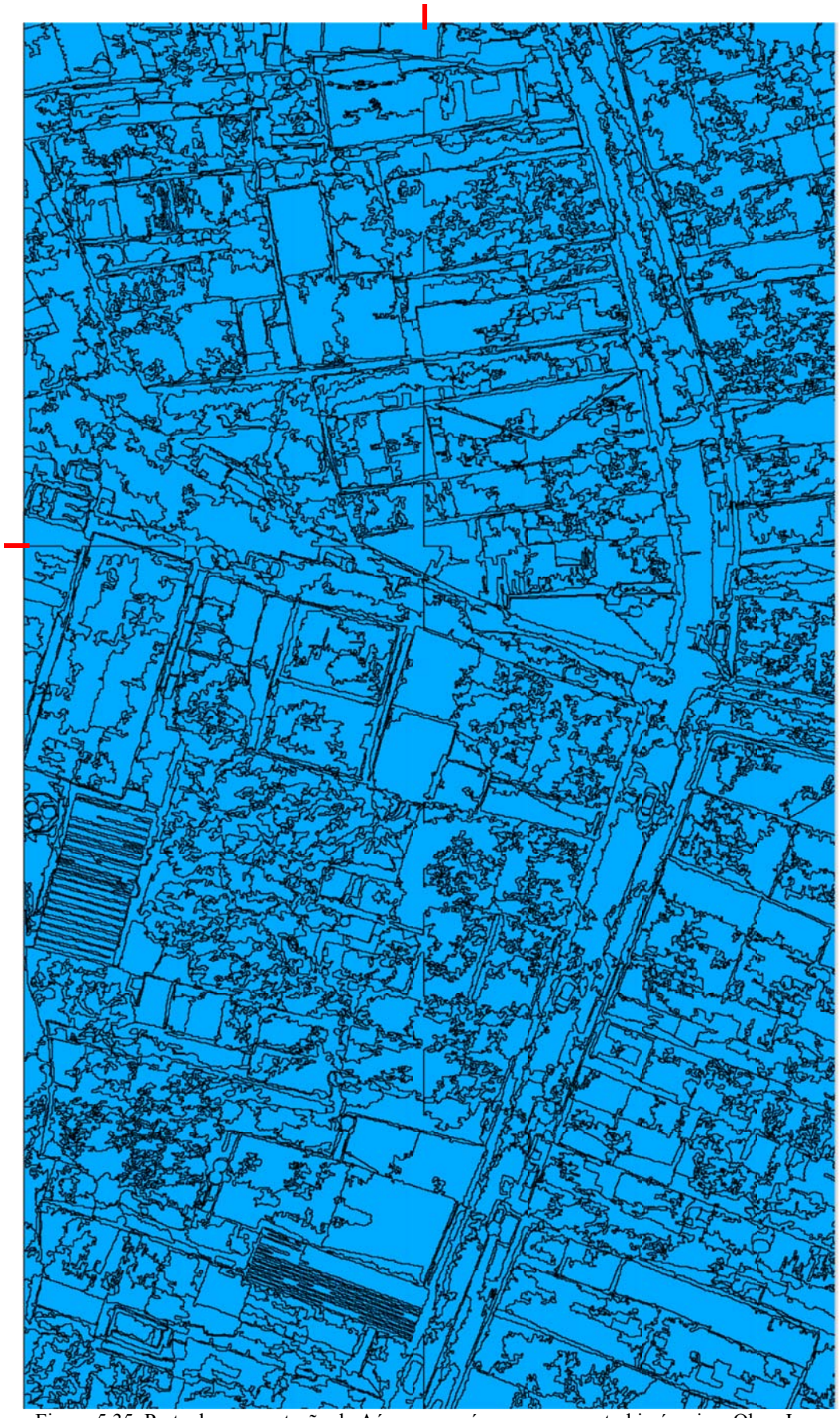
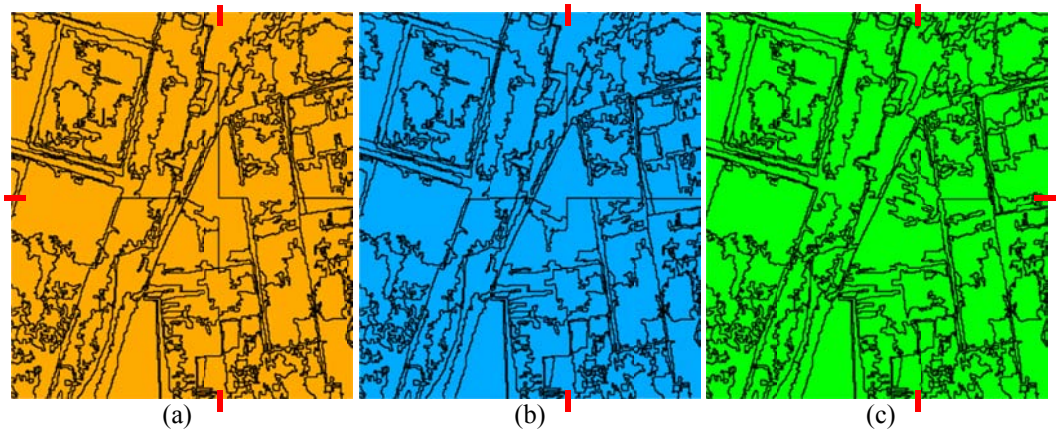


Figura 5.35. Parte da segmentação de Aérea com pós-processamento hierárquico. Obs.: Imagem rotacionada.

Figura 5.36. Parte de Aérea com pós-processamento hierárquico com ressegmentação. Obs.: Imagem rotacionada.



(a) (b) (c)

Figura 5.37. Comparação entre as estratégias de pós-processamento: simples (a), hierárquico (b) e hierárquico com segmentação (c) para imagem Aérea.

6
Conclusão

Esta tese introduz um método para segmentação de imagens distribuída na nuvem de forma a possibilitar o processamento de imagens grandes de altíssima resolução, ou seja, um grande volume de dados que não pode ser executado em uma única máquina. A solução proposta se baseia no modelo MapReduce, porém as ideias centrais não se limitam a este paradigma. As plataformas Hadoop e Pig foram utilizadas para realizar a execução distribuída, no entanto a escolha de outras plataformas também são válidas.

O emprego da computação em nuvem faz com que a solução seja acessível tanto pelo aspecto financeiro quanto prático, pois basta que o usuário reserve uma quantidade de máquinas de acordo com a necessidade.

O método apresentado consiste na divisão da imagem em *tiles* de tamanho pré-definido para a realização de uma segmentação independente com alto grau de paralelismo. A fim de refinar a qualidade da segmentação três métodos de pós-processamento são sugeridos: simples, hierárquico e hierárquico com ressegmentação. Assim, além de definir um modelo para a realização da segmentação distribuída na nuvem, existe uma solução também relacionada com a forma de realizar um pós-processamento distribuído a fim de eliminar artefatos no resultado final.

Dentre os três métodos de pós-processamento apresentados, o primeiro é o que atinge o maior grau de paralelismo alcançando melhor desempenho computacional enquanto o último é o que atinge menor *speedup*. Em contrapartida, a qualidade da segmentação é exatamente o oposto. Sendo assim, é possível escolher entre qualidade e desempenho de acordo com a seleção da etapa de pós-processamento.

Os resultados obtidos demonstram que a solução proposta é eficiente e possui um potencial altamente escalável, uma vez que o *speedup* aumenta de acordo com o número de máquinas utilizado. Os experimentos realizados também

corroboram a ideia de que maiores tamanhos de imagem resultarão em maiores ganhos de desempenho.

Para a segmentação com pós-processamento simples um *speedup* maior que 16 foi alcançado ao processar uma imagem de 32.000 x 32.000 *pixels* utilizando 32 máquinas para processamento. No caso do pós-processamento hierárquico o *speedup* de 11,5 foi obtido sob as mesmas condições.

É importante ressaltar ainda que o método de segmentação distribuída foi desenvolvido em conjunto com a elaboração da plataforma de interpretação de imagens distribuída na nuvem InterIMAGE Cloud Platform (ICP). Na realidade, o ICP (Ferreira et al., 2014) é uma readaptação do sistema original InterIMAGE (Costa et al., 2008) concebido para processar grandes conjuntos de dados na nuvem. A plataforma fornece ferramentas para a interpretação automática de imagens através da definição de modelos e grafos de operadores.

Dentro do conceito do ICP, a segmentação de imagens é de suma importância. Os métodos desenvolvidos nesta tese estão também de acordo com o funcionamento do sistema e podem ser facilmente acoplados como operadores externos.

6.1.Trabalhos futuros

Em relação à continuidade do trabalho, alguns ajustes relacionados à programação ainda se fazem necessários quando se trata da implementação do pós-processamento hierárquico com ressegmentação de forma a garantir que não haja qualquer problema relacionado à memória independentemente do tamanho da imagem.

Adicionalmente, é desejável a realização de experimentos envolvendo uma avaliação sistemática da qualidade e estabilidade da segmentação em relação ao tamanho dos *tiles*. Os resultados da segmentação distribuída também devem ser comparados com os resultados oriundos da versão estritamente sequencial a fim de garantir maior validação de qualidade. Experimentos considerando imagens ainda maiores também são opções interessantes, bem como uma maior variação da infraestrutura utilizada (número de máquinas) e uma análise da utilização da segmentação distribuída dentro de um modelo completo de interpretação desenvolvido no ICP.

Há ainda a possibilidade de adaptar versões paralelas de segmentação de imagens com a finalidade de incluir um novo grau de paralelismo dentro de cada unidade de execução do processamento distribuído. Adicionalmente, um cluster de GPUs pode ser uma alternativa para aumentar ainda mais o desempenho deste tipo de execução.

Também com o intuito de prover maior desempenho, há de se pensar em um gerenciamento inteligente para a utilização das máquinas que ficam ociosas durante algumas etapas de pós-processamento hierárquico e hierárquico com ressegmentação.

É desejável também a realização de experimentos envolvendo a utilização do Spark (Apache Spark, 2015) que pode ser executado em combinação com o Hadoop de forma a aproveitar o sistema de arquivos distribuído existente. Para determinadas aplicações, Spark pode alcançar um desempenho até 100 vezes maior do que o Hadoop MapReduce. Neste sentido, a opção pela utilização de scripts Pig Latin na implementação do método proposto foi particularmente vantajosa, uma vez que podem ser prontamente adaptados para trabalhar com Spark com a mudança de apenas uma linha de código.

Outro trabalho interessante está relacionado com a implementação do método proposto para outros algoritmos de crescimento de regiões ou até outras técnicas de segmentação demonstrando sua adaptabilidade. A utilização de imagens hiperespectrais também apresentam uma boa possiblidade de análise experimental.

Por fim, outras abordagens para segmentação distribuída também carecem de ser investigadas. A primeira vertente tem a ver com o emprego de um pré-processamento inteligente de forma a eliminar as onerosas etapas de pós-processamento. Este pré-processamento está relacionado com a divisão em *tiles* de forma adaptativa, utilizando critérios de similaridade para definir as fronteiras dos *tiles*. Desta forma, o custo computacional da etapa de divisão da imagem seria mais elevado, porém o resultado final poderia ser produzido pela segmentação independente de forma satisfatória, excluindo assim a etapa de pós-processamento. Uma ideia similar é apresentada em (Korting et al., 2013).

A outra ideia está relacionada com à utilização de *tiles* com sobreposição de forma a produzir áreas estáveis onde há a garantia de que, para uma dada quantidade de iterações, o crescimento de regiões será sempre igual. Esta solução

remete à margem de estabilidade proposta em (Lassale et al., 2014) e requer que após uma quantidade pré-determinada de iterações, as iterações restantes do crescimento de regiões sejam realizadas uma de cada vez como processos distribuídos.

Referências bibliográficas

ACETO, G.; BOTTA, A.; DONATO, W.; PESCAPÈ, A. **Cloud monitoring: A survey**, Computer Networks, Volume 57, Issue 9, 19 June 2013, Pages 2093-2115

ACHANCCARAY, P.; AYMA, V.; JIMENEZ, L.; GARCIA, G. S.; HAPP, P. N.; FEITOSA, R. Q.; PLAZA, A. **A free software tool for automatic tuning of segmentation Parameters**. South-Eastern European Journal of Earth Observation and Geomatics, Special-Thematic Issue, GEOBIA 2014: Advancements, trends and challenges, 5th Geographic Object-Based Image Analysis Conference, Thessaloniki, Greece, May 21-24, 2014, Vo3, No2S, p. 707-711, 2014.

AL-HAMEEDAWI, A.; BUCHROITHNER, M. **Object-oriented classifications for land use/land cover using Cosmo-SkyMed and LandSat 7 satellite data: An example of Erbil/Iraq**, EUSAR 2014; 10th European Conference on Synthetic Aperture Radar; Proceedings of , vol., no., pp.1,4, 3-5 June 2014.

AMAZON WEB SERVICES. **Landsat on AWS**. http://aws.amazon.com/pt/public-data-sets/landsat/ (acessado em julho de 2015).

AMAZON WEB SERVICES. http://aws.amazon.com/pt/ (acessado em julho de 2015).

APACHE HADOOP. http://hadoop.apache.org (acessado em julho de 2015).

APACHE PIG. http://pig.apache.org/ (acessado em em julho de 2015).

APACHE SPARK. http://spark.apache.org/ (acessado em em julho de 2015).

APOGEO SPATIAL. **WorldView-3! - Seeing Through Smoke, Into Water and Earth**, Outubro de 2014. http://apogeospatial.com/worldview-3/ (acessado em julho de 2015).

ASANOVIC, K. et Al. **A view of the parallel computing landscape**. Communications of the ACM, 52(10):56-67, 2009.

ASSUNÇÃO, M. D. CALHEIROS, R. N. BIANCHI, S. NETTO, M. A. S. BUYYA, R. **Big Data computing and clouds: Trends and future directions**, Journal of Parallel and Distributed Computing, Agosto de 2014.

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.

BACKER, M.; TÜNNERMANN, J.; MERTSCHING, B. **Parallel K-Means Image Segmentation Using Sort, Scan and Connected Components on a GPU**. In Facing the Multicore-Challenge III, 108–120. Berlin: Springer, 2013.

BAGGIO, D. L. GPGPU Based Image Segmentation Livewire Algorithm Implementation. Aeronautical Institute of Technology, 2007, São José dos Campos, 2007. (Dissertação de Mestrado)

BEUCHER, S.; MEYER, F. **The morphological approach to segmentation: the watershed transformation in Mathematical Morphology in Image Processing**, E. R. Dougherty, Editor, pp. 433-481, Marcel Dekker Inc., 1993.

BLASCHKE, T., 2010, **Object based image analysis for remote sensing**, ISPRS Journal of Photogrammetry and Remote Sensing, 65 (1), January 2010, pp. 2-16.

BLASCHKE, T. et Al. **Geographic Object-Based Image Analysis - Towards a new paradigm**. ISPRS Journal of Photogrammetry and Remote Sensing., v.87, p.180 - 191, 2014.

BLASCHKE, T.; STROBL, J. **What is wrong with pixels? Some recent developments interfacing remote sensing and GIS**. GIS-Zeitschrift für Geoinformationssysteme, n. 6, p. 12-17, 2001.

BREITMAN, K. K.; VITERBO, J. **Computação na Nuvem - Uma Visão Geral**. CONSEGI 2010 – III Congresso Internacional Software Livre e Governo Eletrônico. Brasília, 2010. p. 23-44.

BYRD, K.; JIANCHAO ZENG; CHOUIKHA, M. **Performance assessment of mammography image segmentation algorithms**, Applied Imagery and Pattern Recognition Workshop, 2005. Proceedings. 34th , vol., no., pp.6 pp.-157, 1-1, 2005.

CARDOSO, J. S.; CORTE-REAL, L. **Toward a generic evaluation of image segmentation**, IEEE Transactions on Image Processing 14 (11), 2005.

COMANICIU, D.; MEER, P. **Mean shift: A robust approach toward feature space analysis.** IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 24, pp. 603-609, 2002.

CORREIA, P.; PEREIRA, F. **Objective evaluation of relative segmentation quality**, Image Processing, 2000. Proceedings. 2000 International Conference on , vol.1, no., pp.308-311 vol.1, 2000.

DEAN, J; GHEMAWA, S. **MapReduce: Simplified Data Processing on Large Clusters**. Google Labs, OSDI; 2004 137–150.

CANNY, J. **A Computational Approach To Edge Detection**, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

COSTA, G. A. O. P.; PINHO, C. M. D.; FEITOSA, R. Q.; ALMEIDA, C. M.; KUX, H. J. H.; FONSECA, L. M. G.; OLIVEIRA, D. A. B.. **InterIMAGE: uma Plataforma Cognitiva Open Source para a Interpretação Automática de Imagens Digitais.** Revista Brasileira de Cartografia, SBC – Sociedade Brasileira de Cartografia, Geodésia, Fotogrametria e Sensoriamento Remoto, Rio de Janeiro, v. 60(4), p. 331-337, 2008.

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 , pp.184-189, 2008.

DEY, V.; ZHANG, Y. M.; ZHONG, M. **A review on image segmentation techniques with remote sensing perspective**. In: e, in Proceedings of the International Society for Photogrammetry and Remote Sensing Symposium (ISPRS10), vol. XXXVIII (Part 7A), Austria, July 5–7, 2010.

DIKAIAKOS, M.D.; KATSAROS, D.; MEHRA, P.; PALLIS, G.; VAKALI, A. **Cloud Computing: Distributed Internet Computing for IT and**

Scientific Research, Internet Computing, IEEE , vol.13, no.5, pp.10,13, Setembro-Outubro, 2009.

EARTHDATA. **Getting Petabytes to People: How the EOSDIS Facilitates Earth Observing Data Discovery and Use**. https://earthdata.nasa.gov/getting-petabytes-to-people-how-the-eosdis-facilitates-earth-observing-data-discovery-and-use (acessado em julho de 2015)

ECOGNITION, http://www.ecognition.com/suite/ecognition-developer (acessado em julho de 2015).

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

FERREIRA, R. S.; OLIVEIRA, D. B.; HAPP, P. N.; COSTA, G. A. O. P.; FEITOSA, R. Q.; BENTES, C. **InterIMAGE 2: The architecture of an open-source, high performance platform for automatic, knowledge-based image interpretation**. South-Eastern European Journal of Earth Observation and Geomatics, Special-Thematic Issue, GEOBIA 2014: Advancements, trends and challenges, 5th Geographic Object-Based Image Analysis Conference, Thessaloniki, Greece, May 21-24, 2014, Vo3, No2S, p. 275-280, 2014.

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

Gates, A. **Programming Pig**, O'Reilly, 2011.

GEOBIA 2014, Conference proceedings http://geobia2014.web.auth.gr/geobia14/news/conference-proceedings (acessado em julho de 2015).

GILDER, G. **The Information Factories**, Wired Magazine, Outubro, 2006. http://archive.wired.com/wired/archive/14.10/cloudware.html (acessado em julho de 2015).

GISGEORAPHY. **100 Earth Shattering Remote Sensing Applications & Uses**, Abril de 2015. http://gisgeography.com/100-earth-remote-sensing-applications-uses/ (acessado em julho de 2015).

GONZALEZ, R. G.; WOODS, R. E. **Digital Image Processing**; 3rd Edition, Prentice Hall, 2007.

GOOGLE OPEN SOURCE BLOG. **MapReduce for C: Run Native Code in Hadoop**, 2015. http://google-opensource.blogspot.com.br/2015/02/mapreduce-for-c-run-native-code-in.html (acessado em julho de 2015).

HADAMARD, J. **Sur les problèmes aux dérivées partielles et leur signification physique**. Princeton University Bulletin. pp. 49–52, 1902.

HADOOP WIKI. **PoweredBy** http://wiki.apache.org/hadoop/PoweredBy (acessado em julho de 2015).

HALDER, A; GIRI, C.; HALDER, A. **Brain tumor detection using segmentation based Object labeling algorithm**, Electronics, Communication and Instrumentation (ICECI), 2014 International Conference on , vol., no., pp.1,4, 16-17 Jan. 2014.

HAPP, P. N.; FEITOSA, R. Q.; STREET, A. **Assessment of optimization methods for automatic tunning of segmentation parameters**. In: IV International Conference on Geographic Object Based Image Analysis, 2012, Rio de Janeiro. Proceedings of GEOBIA 2012. São José dos Campos: INPE, 2012. v.1. p. 490-495.

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.; FEITOSA, R.Q.; BENTES, C.; FARIAS, R. **A Region-Growing Segmentation Algorithm for GPUs**. Geoscience and Remote Sensing Letters, IEEE , vol.10, no.6, pp.1612,1616, Nov. 2013.

HARALICK, R. M.; SHAPIRO, L. G. **Computer and Robot Vision**. Addison-Wesley Publishing Company, Volume 1, 1992.

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

HASHEM, I. A. T.; YAQOOB, I.; ANUAR, N. B.; MOKHTAR, S.; GANI, A.; KHAN, S. U. **The rise of 'big data' on cloud computing: Review and open research issues**, Information Systems, Volume 47, Janeiro de 2015, 98-115.

HAY, G. J. **GEOBIA – Evolving Beyond Segmentation**. GEOBIA 2014: Keynote Presentation, 5th Geographic Object-Based Image Analysis Conference, Thessaloniki, Greece, May 21-24, 2014.

HAY, G. J.; CASTILLA, G. **Geographic Object-Based Image Analysis (GEOBIA): A new name for a new discipline**. em Blaschke, T., Lang, S., Hay, G. (Eds.), Object Based Image Analysis. Springer, Heidelberg, Berlin, New York, 2008, pp. 93–112.

HEIMANN, T. et al. **Comparison and Evaluation of Methods for Liver Segmentation From CT Datasets**, Medical Imaging, IEEE Transactions on , vol.28, no.8, pp.1251-1265, Ago. 2009.

HITCHINGS, S. **Policy assessment of the impacts of remote-sensing technology**. Space Policy 19:119 -125, 2013.

HOLMES, A. **Hadoop in Practice**. Manning Publications Co., Shelter Island, New York, 2012.

INTERIMAGE. http://www.lvc.ele.puc-rio.br/projects/interimage/pt-br/ (acessado em julho de 2015)

JIN, X.; SIEBER, R.; KALACSKA, M. **The challenges of image segmentation in big remotely sensed imagery data**. Annals of GIS 20.4 (2014): 233-244, 2014.

KORTING, T. S.; CASTEJON, E. F.; FONSECA, L. M. G. The Divide and Segment Method for Parallel Image Segmentation., in Jacques Blanc-Talon; Andrzej J. Kasinski; Wilfried Philips; Dan C. Popescu & Paul Scheunders, ed., Advanced Concepts for Intelligent Vision Systems , Springer, , pp. 504-515, 2013.

LASSALE, P.; INGLADA, J.; MICHEL, J.; GRIZNONNET, M.; MALIK, J. **Large scale region-merging segmentation using the local mutual best fitting concept.** Geoscience and Remote Sensing Symposium (IGARSS), 2014 IEEE International , vol., no., pp.4887,4890, 13-18 July 2014

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-July 3, 2009.

JSON. http://json.org/ (acessado em julho de 2015).

MACQUEEN, J. B. **Some Methods for classification and Analysis of Multivariate Observations**. in Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probabilit, University of California Press, pp.281–297, 1967.

MARPU, P. R.; NEUBERT, M.; HEROLD, H.; NIEMEYER, I. **Enhanced evaluation of image segmentation results**. Journal of Spatial Science vol. 55, Issue 1, 2010 pp.56-58.

MEI-PING SONG; XING-WEI XU; SHUANGYANG LU; WEI XU; HAI-MO BAO. **Applying region growing algorithm to hyperspectral image for oil segmentation**, Proc. SPIE 9124, Satellite Data Compression, Communications, and Processing X, 912404, May, 2014.

MELL, P.; GRANCE, T. **The NIST Definition of Cloud Computing**. SP 800-145 - Technical Report, National Institute of Standards & Technology Gaithersburg, MD, United States, 2011.

MICHEL, J.; GRIZONNET, M.; JAEN, A.; HARASSE, S.; HERMITE, L.; GUINET, J.; MALIK, J.; SAVINAUD, M. **Open tools and methods for large scale segmentation of very high resolution satellite images**, in Open Source and Geospatial Research and Education Symposium, 2012, pp. 179–184, 2012.

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.

NARKHEDE, H.P. **Review on Image Segmentation Techniques**, International Journal of Science and Modern Engineering, Vol 1, Issue 8, Julho de 2013.

NETWORK COMPUTING. **How DigitalGlobe Handles 2 Petabytes Of Satellite Data Yearly**, Dezembro de 2013. http://www.networkcomputing.com/cloud-infrastructure/how-digitalglobe-handles-2-petabytes-of-satellite-data-yearly/d/d-id/1113278 (acessado em julho de 2015).

NEUBERT, M., MEINEL, G. **Evaluation of segmentation programs for high resolution remote sensing applications**. In: Schroeder, M., Jacobsen, K., Heipke, C. (Eds.). Proceedings Joint ISPRS/EARSeL Workshop "High Resolution Mapping from Space 2003", Hannover, Germany, October 6-8, 2003.

NEUBERT, M.; HEROLD, H.; MEINEL, G. **Accessing image segmentation quality – concepts, methods and application** In: BLASCHKE, T.; LANG, S.; HAY, G. (Eds.). Object-Based Image Analysis: Spatial concepts for knowledge-driven remote sensing applications. Heidelberg: Springer, 2008. cap. 8.3, p. 679-784.

NOVACK, T.; KUX, H. J. H. **Urban land cover and land use classification of an informal settlement area using the open-source knowledge-based system InterIMAGE**. Journal of Spatial Science, 55: 1, 23- 41, 2010.

OPEN GEOSPATIAL CONSORTIUM. **Geographic information - Well-known text representation of coordinate reference systems**. http://docs.opengeospatial.org/is/12-063r5/12-063r5.html (acessado em julho de 2015)

OPEN FORUM. **Skybox, Big Geospatial Data**, Abril de 2015. https://openforum.hbs.org/challenge/understand-digital-transformation-of-business/data/skybox-big-geospatial-data (acessado em julho de 2015).

OWENS, J. D.; HOUSTON, M.; LUEBKE, D.; GREEN, S.; STONE, J. E.; PHILLIPS, J. C. **GPU Computing**, Proceedings of the IEEE , vol.96, no.5, pp.879-899, May 2008.

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

PALOMERA-PEREZ, M.A.; MARTINEZ-PEREZ, M.E.; BENITEZ-PEREZ, H.; ORTEGA-ARJONA, J.L. **Parallel Multiscale Feature Extraction and Region Growing: Application in Retinal Blood Vessel Detection**.

Information Technology in Biomedicine, IEEE Transactions on, vol.14, no.2, pp.500-506, março 2010.

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.

PAVLIDIS, T. **Image analysis**, Annual Review of Computer Science, 1988.

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

PIGNALBERI, G.; CUCCHIARA, R.; CINQUE, L.; LEVIALDI, S., 2003. **Tuning range image segmentation by genetic algorithm**. EURASIP Journal on Applied Signal Processing, n. 8, p. 780–790.

PONT-TUSET, J.; MARQUES, F. **Measures and Meta-Measures for the Supervised Evaluation of Image Segmentation**. IEEE Conference on Computer Vision and Pattern Recognition- CVPR. June 23-28, 2013, pp. 2131-2138.

PREWITT, J. **Object Enhancemet And Extraction**. in Picture Processing and Psychopictorics (B. Lipkin and A. Rosenfeld, Ed.), NY, Academic Pres, 1970.

REUTERS. **DigitalGlobe gains U.S. govt license to sell sharper satellite imagery**, Junho de 2014. http://www.reuters.com/article/2014/06/11/digitalglobe-imagery-idUSL2N0OR2UX20140611 (acessado em julho de 2015).

REUTERS. **DigitalGlobe Announces Availability of 30 cm Satellite Imagery to All Customers**, Fevereiro de 2015. http://uk.reuters.com/article/2015/02/25/co-digitalglobe-idUKnBw255178a+100+BSW20150225 (acessado em julho de 2015).

RISEMAN, E. M.; ARBIB, M. A. **Computational techniques in the visual segmentation of static scenes**, Computer 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.

SAGAN, H. **Space-Filling Curves**. Springer-Verlag, 1994.

SAMPLE, J. T.; IOUP, E. **Tile-Based Geospatial Information Systems – Principles and Practices**. Springer Science+Business Media, LLC, New York, NY, 2010.

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

SINGH, D. E.; HERAS, D. B.; RIVERA, F. F. **Parallel seeded region growing algorithm**. Proc. VIII Simposium Nacional de Reconocimiento de Formas y Analisis de Imagenes. SNRFAI'99. Bilbao(Spain), 1999.

SKARBEK, W.; KOSCHAN, A. **Colour image segmentation – a survey**, Technical Report, Tech. Univ. of Berlin, outubro 1994.

SNAPPY. https://code.google.com/p/snappy/ (acessado em julho de 2015).

SOBEL, I., **An Isotropic 3×3 Gradient Operator**, Machine Vision for Three – Dimensional Scenes, Freeman, H., Academic Pres, NY, 376-379, 1990.

SOSINSKY, B. **Cloud Computing Bible**. Wiley Publishing, Inc., Indiana, Indianapolis, 2011.

SPACE FLIGHT NOW. **Google buys startup Skybox Imaging for $500 million**, 2014. http://www.spaceflightnow.com/news/n1406/10google/#.Vawso_lVhBd (acessado em julho de 2015).

STALLINGS, W. **Computer Organization and Architecture: Design for Perfomance**. 8th Ed., Pearson, 2009.

SURVE, A. R.; PADDUNE, N. S. **A Survey on Hadoop Assisted K-Means Clustering of Hefty Volume Images**. International Journal on Computer Science and Engineering 6.3 (2014): 113, 2014.

TESFAMARIAM, E. B. **Distributed processing of large remote sensing images using MapReduce - A case of Edge Detection**. Institute for Geoinformatics Universität Münster, Germany, 2011. (Dissertação de Mestrado)

VANTARAM, S. R.; SABER, E. **Survey of contemporary trends in color image segmentation**. Journal of Electronic Imaging. 2012, Vol. 21, 4, pp. 040901-1-040901-28.

VAQUERO, L. M.; RODERO-MERINO, L.; CACERES, J.; LINDNER, M. **A break in the clouds: towards a cloud definition**. ACM SIGCOMM Computer Communication Review 39 (1), 50-55, 2009.

VATSAVAI, R. R. **Object based image classification: state of the art and computational challenges.** In Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data (BigSpatial '13). ACM, New York, NY, USA, 73-80, 2013.

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.

WHITE, T. **Hadoop: The Definitive Guide**. O'Reilly, 2012.

XIN CAO; QIANGZI LI; XIN DU; MIAO ZHANG; XINQI ZHENG. **Exploring effect of segmentation scale on orient-based crop identification using HJ CCD data in Northeast China**. IOP Conference Series: Earth and Environmental Science Volume 17, conference 1, 2014.

XVII SBSR. http://www.dsr.inpe.br/sbsr2015/ (acessado em julho de 2015).

YAHOO!, **Apache Hadoop Tutorial** http://developer.yahoo.com/hadoop/tutorial/module1.html (último acesso em julho de 2014).

YAN XU; JUN-YAN ZHU; ERIC I-CHAO CHANG; MAODE LAI; ZHUOWEN TU; **Weakly supervised histopathology cancer image segmentation and classification**, Medical Image Analysis, Volume 18, Issue 3, April 2014, Pages 591-604.

ZHANG, H.; FRITTS, J. E.; GOLDMAN, S. A. I**mage segmentation evaluation: A survey of unsupervised methods**, Computer Vision and Image Understanding, Volume 110, Issue 2, May 2008, Pages 260-280.

ZHANG, Y. J. **A survey on evaluation methods for image segmentation**, Pattern Recognition, Volume 29, Issue 8, August 1996, Pages 1335-1346