EXTRAÇÃO DE MODELO DIGITAL DE ELEVAÇÕES ACELERADA EM GPU

Irving da Silva Badolato

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Ricardo Cordeiro de Farias

Rio de Janeiro
Setembro de 2014

EXTRAÇÃO DE MODELO DIGITAL DE ELEVAÇÕES ACELERADA EM GPU

Irving da Silva Badolato

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof. Ricardo Cordeiro de Farias, Ph.D.

_____

Prof. Franklin de Lima Marquezino, D.Sc.

_____

Prof. Cristiana Barbosa Bentes, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2014

Badolato, Irving da Silva

Extração de Modelo Digital de Elevações Acelerada em GPU/Irving da Silva Badolato. – Rio de Janeiro: UFRJ/COPPE, 2014.

XII, 54 p.: il.; 29, 7cm.

Orientador: Ricardo Cordeiro de Farias

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2014.

Referências Bibliográficas: p. 51 – 54.

1. Modelo Digital de Elevações. 2. Reconstrução de Cena. 3. Programa em GPU. I. Farias, Ricardo Cordeiro de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedico esta dissertação à minha noiva
e à memória de meu avô materno.

Agradecimentos

À minha querida noiva, Michelle Costa, pelo carinho, confiança, apoio e críticas, pois suas contribuições trouxeram o fortalecimento nos momentos difíceis. E também pelas figuras em perspectiva usadas neste trabalho, pois são frutos de sua dedicação com o desenho.

À toda minha família, pois ali estão os tios e tias que são referência em minha vida e residem as lembranças de meu querido avô Angelino A. Antônio.

Ao meu orientador Ricardo Farias (Ph.D.) pela compreensão e confiança que junto de seus ensinamentos e orientações permitiram a realização deste projeto.

À amiga e colega de formação em graduação e mestrado Sarina Lustosa (M.Sc.), ao professor Ricardo Marroquim (D.Sc.) e aos demais amigos e professores do PESC/COPPE, pelo convívio e conhecimentos compartilhados.

À equipe de secretariado do programa, em especial ao Gutierrez da Costa, pela paciência e empenho em ajudar na superação dos obstáculos burocráticos no curso.

Ao coordenador do Projeto E-Foto Jorge Nunes (Ph.D.) e aos professores de graduação Cristiana Bentes (D.Sc.), Guilherme Abelha (D.Sc.), João Araújo (D.Sc.) e Orlando Bernardo (D.Sc.), pelos ensinamentos e pela motivação para a pesquisa científica.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

EXTRAÇÃO DE MODELO DIGITAL DE ELEVAÇÕES ACELERADA EM GPU

Irving da Silva Badolato

Setembro/2014

Orientador: Ricardo Cordeiro de Farias

Programa: Engenharia de Sistemas e Computação

Apresenta-se, nesta dissertação, uma proposta para aceleração do processo de aquisição automática de Modelos Digitais de Elevação. Como referência é adotado um algoritmo preexistente que está disponível segundo licença GNU GPL no e-foto, um software aplicado à educação em fotogrametria digital e mapeamento topográfico. Primeiramente são abordadas técnicas de criação de perfis para análise de performance das chamadas de funções e cobertura de código para verificar o esforço computacional e determinar gargalos. Em seguida é descrita otimização para melhoria do algoritmo de correlação cruzada normalizada utilizado em conjunto com a técnica de crescimento de regiões para buscar pontos homólogos. Pontos homólogos são os pontos utilizados na determinação das coordenadas tridimensionais viabilizando a reconstrução de cena aplicada aos pares de imagens estereoscópicas. Finalmente, é elaborado o conjunto de rotinas de um programa em GPU e são realizados testes para determinar o ganho em relação a versão original do programa que executa sequencialmente em CPU.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

GPU-ACCELERATED DIGITAL ELEVATION MODEL EXTRACTION

Irving da Silva Badolato

September/2014

Advisor: Ricardo Cordeiro de Farias

Department: Systems Engineering and Computer Science

It is presented in this work a proposal to speed up the automatic acquisition of Digital Elevation Models. Is adopted as a reference a pre-existing algorithm that is available under GNU GPL on e-foto, a software applied to education in digital photogrammetry and topographic mapping. First, techniques of profiling for performance analysis in function calls and source code coverage are discussed to verify the computational effort and determine bottlenecks. Then, is described a optimization to improve the normalized cross-correlation algorithm used along with the region growing technique for searching homologous points. Homologous points are the points used in determining the three-dimensional coordinates enabling the scene reconstruction applied to pairs of stereoscopic images. Finally, a set of routines is prepared to GPU program and tests are conducted to determine the gain compared to the original version of the program running sequentially on the CPU.

Sumário

Lista de Figuras **x**

Lista de Tabelas **xii**

1 Introdução **1**

1.1 Considerações Gerais 1
1.2 Motivações 4
1.3 Proposta de Trabalho 5
1.4 Organização da Dissertação 6

2 Revisão Bibliográfica **7**

2.1 Extração de Modelos Digitais de Elevação 7
2.1.1 Orientação de Imagens 7
2.1.2 Busca de Homólogos 11
2.1.3 Triangulação 13
2.2 Arquiteturas GPU 14
2.2.1 Visão Global 14
2.2.2 Conceitos 16

3 Método Proposto **19**

3.1 Algoritmo Sequencial 19
3.1.1 Análise com *Gprof* e *Gcov* 22
3.2 Otimização 26
3.3 Algoritmo Paralelo 30
3.3.1 Divisão do Processamento 31
3.3.2 Nova Estratégia de Crescimento de Regiões 32

4 Resultado e discussões **39**

4.1 Ambiente de Teste e Dados Utilizados 39
4.2 Tamanho da Janela de Busca 41
4.3 Redução de Chamadas de Função 41
4.4 Análise de Cobertura 43

4.5 Qualidade dos Resultados 48
4.6 Tempos de Execução 49

5 Conclusão e Trabalhos Futuros **50**

Referências Bibliográficas **51**

Lista de Figuras

1.1 A captura de um ponto em imagens estereoscópicas 2
1.2 Diferentes representações para modelos digitais de elevação 3

2.1 Condição de colinearidade 9
2.2 Workflow de uma estação fotogramétrica digital 10
2.3 Conceitos de template, imagem, janela de busca e amostra de maior correlação 11
2.4 Triangulação de coordenadas 14
2.5 Como GPUs podem acelerar o processamento 15
2.6 Modelo de distribuição de processos para execução em GPU 17
2.7 Redução de 8 elementos 18

3.1 Diagrama de sequência para o programa preexistente 22
3.2 Grafo de chamadas do programa preexistente 23
3.3 Gráfico de frequência do uso para algumas das linhas de código do programa preexistente 24
3.4 Fluxo geral de execução 31
3.5 Ilustração da fronteira do crescimento de regiões 36
3.6 Ilustração do mapa de controle de regiões visitadas 37
3.7 Casos de colisão de fronteiras de crescimento 38

4.1 Par de imagens estereoscópicas (1/2) e distribuição dos pontos usados como sementes nos testes 40
4.2 Par de imagens estereoscópicas (2/2) e distribuição dos pontos usados como sementes nos testes 40
4.3 Par de imagens estereoscópicas após atuação da correção radiométrica de aproximação do histograma das imagens 40
4.4 Variação de tempos de execução e número de pontos obtidos em função do tamanho das janelas de busca 41
4.5 Grafo de chamadas após modificação do programa preexistente 43
4.6 Comparativo da cobertura em edificações muito elevadas ou cercadas por regiões de difícil correlação 44

4.7 Evolução do crescimento de regiões no programa preexistente 46
4.8 Evolução do crescimento de regiões na solução proposta 47

Lista de Tabelas

3.1 Perfil do tempo de execução para o programa preexistente 23
3.2 Variáveis adotadas no código otimizado 27

4.1 Comparação dos tempos de execução para extrair o modelo de um par de imagens entre o programa preexistente inalterado e com modificações 42
4.2 Perfil do tempo de execução do programa preexistente com modificações de acesso direto à memória 42
4.3 Comparação do número de buscas de homólogos disparadas por cada semente para o programa preexistente e a solução proposta 45
4.4 Qualidade dos resultados 48
4.5 Comparação dos tempos de execução e determinação da aceleração obtida entre o programa preexistente e a solução proposta 49
4.6 Comparação dos tempos de execução e determinação da aceleração obtida entre os modos de execução da solução proposta 49

Capítulo 1

Introdução

1.1 Considerações Gerais

Modelos de representação da superfície terrestre podem ser nomeados de diferentes formas[1] de acordo com a definição sobre o que é a superfície de interesse. Isto é exemplificado pelo conjunto de definições:

- Modelo Digital de Superfície (DSM - *Digital Surface Model*) termo aplicado a modelos que representam a superfície visível, ou seja, a terra incluindo todos os objetos dispostos sobre ela;
- Modelo Digital de Terreno (DTM - *Digital Terrain Model*) termo aplicado a modelos que representam a superfície de terra nua, ou seja, sem quaisquer objetos como vegetação e construções.
- Modelo Digital de Elevações (DEM - *Digital Elevation Model*) - termo aplicado a modelos que contenham dados altimétricos de uma região específica comumente amostrados a intervalos regulares facilitando assim sua representação em formatos *raster* ou vetoriais.

Na literatura científica não existe consenso para tal terminologia sendo este apenas o caso mais comum. O que, por sua vez, permite a inserção do termo Modelo Digital de Elevações, ou simplesmente MDE, como termo genérico devido ao uso frequente pelos provedores de dados espaciais para designar mapeamentos tridimensionais sem restringir-se a definições da composição da superfície terrestre.

Atribui-se ao professor Charles L. Miller do *Massachusetts Institute of Technology* (MIT) o título de pioneiro no desenvolvimento de técnicas para extração de modelos digitais da superfície terrestre devido a suas publicações[2, 3] dos anos de 1957 e 1958. Por suas contribuições, seus métodos de análise computacional para auxiliar projetos de engenharia civil na elaboração do traçado de estradas foram reconhecidos

no ano de 1998 pela Sociedade Americana de Engenheiros Civis (ASCE - *American Society of Civil Engineers*) com o prêmio *Computing in Civil Engineering Award*[4].

Em sua proposta, as técnicas de modelagem permitiam o adensamento em laboratório do número de pontos de terreno conhecidos utilizando-se de métodos fotogramétricos para mensurar a altimetria em pontos estratégicos resultando numa massa de pontos tridimensionais representando a região de interesse analisada. Nos anos iniciais de produção estes modelos eram obtidos por conjuntos de equipamentos óptico-mecânicos para análise de imagens estereoscópicas e perfuradoras de cartões para promover a entrada de coordenadas e parâmetros para o processamento computacional. Posteriormente, com a evolução da computação e o advento dos sensores CCD[5] (*Charge-Coupled Device*) as partes mecânicas foram substituídas e o processamento das imagens estereoscópicas tornou-se possível totalmente por computador.

O método fotogramétrico[6] para a extração de um MDE utiliza-se do traçado dos feixes luminosos que passam pelo centro óptico de um sistema de câmeras quando este registra um ou mais pares de imagens. A figura 1.1 ilustra os feixes de um mesmo ponto sendo registrados num par de imagens. Na geometria epipolar[7] encontram-se os fundamentos matemáticos que relacionam pontos no espaço tridimensional com suas respectivas projeções em pares de imagens estereoscópicas, permitindo assim, mensurar coordenadas com significativa precisão.

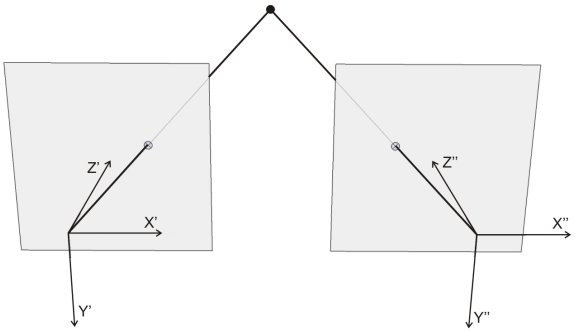

Figura 1.1: A captura de um ponto em imagens estereoscópicas.

Na atualidade modelos digitais de elevações passaram a ser extraídos para diversas aplicações variando-se o objeto de análise. Dentre as aplicações estão:

- A documentação de sítios ou edificações em cartografia, arqueologia e arquitetura para caráter de reconstrução ou preservação;
- A detecção objetos de interesse para veículos autômatos ou robôs industriais;
- A reconstituição e/ou mensuração na perícia criminal ou de acidentes;
- A produção de conteúdo para entretenimento, tais como, filmes e jogos.

Por consequência da diversidade de aplicações existente, há um grande número de formatos de arquivos para armazenamento de um MDE. Entre eles, no segmento de mapeamento topográfico, estão os formatos USGS[8], DTED[9] e DIMAP[10]. Tendo em vista as estruturas de dados suportadas por tais formatos é aceitável tratar a possibilidade de diversas representações dos resultados de uma extração conforme exemplifica a lista a seguir:

- Uma listagem sequencial da nuvem de pontos tridimensionais (fig.1.2-a);
- Uma imagem de profundidade, onde a escala de cores utilizada representa a variação altimétrica da região modelada (fig.1.2-b);
- Uma malha, seja ela regular ou irregular, ou conjunto de vetores que representam a região modelada (fig.1.2-c);
- Uma renderização temática para observação humana de informações adicionais projetadas sobre a região modelada (fig.1.2-d).

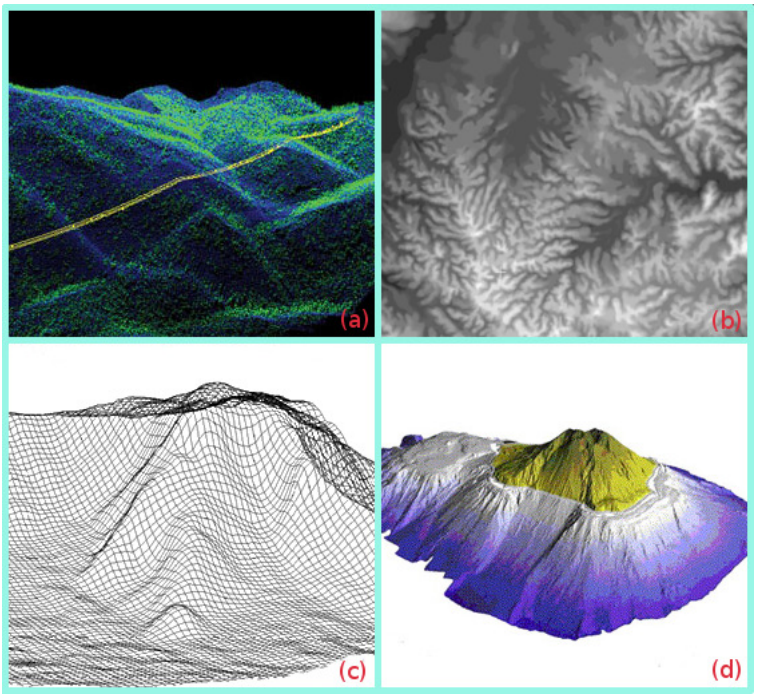

Figura 1.2: Exemplos de representações para Modelos Digitais de Elevações: (a) Nuvem de pontos; (b) Imagem de profundidade; (c) Malha regular; (d) Renderização temática de inundação.

A crescente resolução dos sensores imageadores, aliada a melhorias consideráveis da disponibilidade de memória e poder de processamento computacional, instigam a aplicação destas técnicas em grandes mosaicos e com níveis de detalhamento antes

impraticáveis. Contudo, o balanceamento entre qualidade e tempo de resposta ainda é requisito fundamental a ser considerado nestas aplicações.

Observa-se que a maior parte do poder de processamento dispendido neste problema não está na determinação das coordenadas tridimensionais. O maior custo computacional está na determinação automática dos pontos homólogos, ou seja, o problema envolve analisar segundo uma precisão parametrizável, um ou mais pares estereoscópicos de imagens, afim de determinar por semelhança quais pares de coordenadas tiveram origem na mesma região do espaço tridimensional.

Um operador matemático utilizado para determinar semelhanças em imagens é a Correlação Cruzada Normalizada[11] (NCC - *Normalized Cross-Correlation*) ou índice de correlação de Pearson. Por intermédio deste índice é possível determinar pares de coordenadas das imagens onde os níveis de correlação atendem ao esperado pelo processo. Contudo, isto requer a análise das vizinhanças de cada par distinto de coordenadas, o que por sua vez, aumenta radicalmente o custo computacional a medida em que a área de análise também aumenta.

Uma vez que o valor de correlação pode ser dito independente dos demais valores calculados em diferentes pares de coordenadas, quando respeitada uma distância mínima equivalente a dimensão da vizinhança considerada na análise para cada par, então existe viabilidade para a melhoraria do desempenho das soluções neste segmento pela introdução de hardware especializado no paralelismo do processamento de dados. Utilizando-se das técnicas de desenvolvimento em arquiteturas de Unidades de Processamento Gráfico de Propósito Geral[12] (GPGPU - *General-Purpose Computing on Graphics Processing Units*) é possível acelerar o processo de extração de MDE reduzindo assim o tempo despendido para entrega de resposta.

1.2 Motivações

Para educadores em engenharia existe o desafio de agregar conhecimento tecnológico e científico sem perder as devidas conexões e atualizações sobre a realidade de mercado tendo em vista o desafio de integração de seus alunos num cenário de altíssima concorrência e a grande velocidade com que o conhecimento se molda nos dias atuais. Para tanto é oportuno apresentá-los as ferramentas e a linguagem de mercado, sem contudo exercer parcialidade na escolha de soluções ou torná-los dependentes e consequentemente meros operadores de um conjunto de software.

Ao mesmo tempo os diversos fornecedores que disponibilizam software para os fins de mapeamento topográfico atribuem alto valor aquisitivo e características de licenciamentos que podem acarretar em dificuldades para a educadores e estudantes neste segmento. Estes fatores perpassam as barreiras de custos, observadas principalmente nos núcleos de ensino público, uma vez que o fundamento matemático

importante ao aprendizado é inúmeras vezes omitido nas entranhas de soluções proprietárias.

Em resposta a estas dificuldades surgem iniciativas como a Estação Fotogramétrica Digital Livre do projeto acadêmico E-Foto[13] disponibilizado pelo Laboratório de Fotogrametria da Faculdade de Engenharia da Universidade Estadual do Rio de Janeiro (FEN/UERJ). Desde o ano de 2004 o projeto disponibiliza um software livre, publicações acadêmicas e apoio a estudantes e pesquisadores sobre a elaboração de mapeamento 3D a partir de imagens aéreas obtidas por câmaras métricas de filme ou por sensores digitais aerotransportados.

O projeto possui caráter interdisciplinar, com colaboradores de matemática, geociências e engenharias de sistemas de computação e cartográfica para prover uma solução multiplataforma e de software livre para estudantes e pesquisadores interessados no tema. Seus princípios ou pilares fundamentais são a liberdade e a autoaprendizagem, ou seja, busca-se disponibilizar materiais para dar a oportunidade de experimentar, aprender e evoluir através do uso do software, da leitura de suas publicações, da análise de seu código fonte ou da modificação e redistribuição de seus componentes para atender a diversas necessidades de inovação científica e tecnológica.

Para tanto o conteúdo disponibilizado está licenciado como GNU GPL, ou seja, permite o uso pleno e modificações por terceiros para uso próprio ou publicação sob mesma licença, mantendo assim crédito e referência aos autores originais.

1.3 Proposta de Trabalho

O presente trabalho irá abordar o algoritmo de reconstrução aplicado a mapeamento topográfico que está disponível sob licença de software livre no e-foto[1]. Seu escopo limita-se às funcionalidades existentes no módulo de extração automática de modelos digitais de elevações. Neste contexto, serão abordadas possíveis melhorias ao algoritmo sequencial e propostas rotinas paralelas para a aceleração da extração utilizando GPU.

O ambiente de desenvolvimento adotado foi um computador com hardware GPU da nVidia[14], executando um sistema operacional Linux[15], usando as ferramentas de medição e análise *Gcov*[16] e *Gprof*[17], a linguagem de programação C++[18] e os recursos disponibilizados pelo *toolkit* CUDA[19] para definição de processos em GPU. Houve acoplamento da biblioteca Qt[20] para atender ao requisito de abertura de imagens suportadas pelo e-foto.

[1]Por padrão usamos caixa alta, em Projeto E-Foto, para destacar o nome do projeto acadêmico. Quando ocorre o uso do nome e-foto, em caixa baixa, há referência trata do software resultante dos esforços do projeto.

1.4 Organização da Dissertação

Os próximos capítulos deste trabalho estão organizados da seguinte forma:

- Capitulo 2 - Revisão Bibliográfica: aponta estudos relacionados e resume os conceitos teóricos fundamentais para realização deste trabalho;
- Capitulo 3 - Método Proposto: descreve os métodos de análise e desenvolvimento da aceleração do algoritmo proposto em GPU;
- Capitulo 4 - Resultado e discussões: apresenta e discute os resultados de testes realizados;
- Capitulo 5 - Conclusão e Trabalhos Futuros

Capítulo 2

Revisão Bibliográfica

Neste capítulo serão revisados à luz da literatura os fundamentos para elaboração do tema extração de modelos digitais de elevação acelerada em GPU. Esta abordagem apresenta os conceitos básicos e trabalhos relacionados ao processo de extração de modelos digitais de elevação e ao desenvolvimento em arquiteturas de GPU.

2.1 Extração de Modelos Digitais de Elevação

O modelo digital de elevações é um produto de técnicas de extração que utilizam diversos fenômenos físicos, principalmente aqueles relacionados a propagação da luz, estudados no contexto de interpretação de imagens e reconstrução de cenas. Abordagens clássicas utilizam-se de imagens ao explorar diferentes fenômenos ópticos como a estereoscopia ou a clinometria com infravermelho, conforme afirma Toutin[21]. Contudo, vale citar que existem demais métodos além dos baseados em imagem como, por exemplo, a modelagem pelo emprego das tecnologias de *Real-Time Kinematic*[22] ou de *Light Detection And Ranging*[23].

O método tipicamente implementado nas estações fotogramétricas digitais fundamenta-se em estereoscopia e consiste basicamente da execução sequencial de três etapas: a orientação das imagens, ou seja, o método requer que parâmetros intrínsecos e extrínsecos de cada imagem sejam determinados (item 2.1.1); a detecção de pontos homólogos nas regiões de sobreposição das imagens (item 2.1.2); a triangulação das coordenadas tridimensionais para cada ponto homólogo identificado (item 2.1.3). No processamento de imagens estereoscópicas há trabalhos de pesquisa recentes[24, 25] que demonstram vantagens na aplicabilidade de GPUs.

2.1.1 Orientação de Imagens

A determinação dos parâmetros intrínsecos e extrínsecos de uma imagem, ou simplesmente orientação de imagem, é realizada com intuito de representar as carac-

terísticas construtivas de uma câmera e estabelecer uma relação entre suas imagens e a cena registrada num determinado sistema de coordenadas. Este sistema de coordenadas pode ser arbitrário para objetos pequenos ou associado ao terreno quando a área registrada é grande, podendo assumir representações cartesianas, elipsoidais ou geodésicas de acordo com a aplicação.

A obtenção de parâmetros intrínsecos ou orientação interior[6] depende da execução de calibração do sistema de câmera e implica no conhecimento de suas características construtivas, tais como, dimensões físicas do sensor e a distância focal utilizada.

Sua formulação genérica presume a existência simultânea de até seis parâmetros para tratar alterações de escalas, rotações, translações e não ortogonalidades de eixos:

$$\begin{cases} x = a_0 + a_1.coluna + a_2.linha \\ y = b_0 + b_1.coluna + b_2.linha \end{cases} \tag{2.1}$$

Isto define uma transformada afim entre coordenadas $(linha, coluna)$ do arquivo de imagem e (x, y) do sensor, ou seja, torna possível a conversão que leva coordenadas de um sistema cuja unidade de medida é o *pixel* para um outro em milímetros ou vice-versa.

Além de preservar a formulação existente à época do advento da fotogrametria digital, a transformação entre *pixel* e milímetros realizada na orientação interior torna mais fácil suprimir erros previsíveis como as distorções ocasionadas pelo sistema de lentes. Um método viável para resolver tal problema é relacionar o posicionamento 3D às suas ocorrências em imagens executando uma sequência de operações:

1. Executar a projeção métrica de 3D para 2D (também chamada de resseção espacial);

2. Subtrair erros sistemáticos normalmente expressos em milímetros;

3. Executar a transformada afim para encontrar o posicionamento equivalente no arquivo de imagem.

Se adicionada à técnica de triangulação, do item 2.1.3, o método completa-se pela possibilidade de inferir posições 3D partindo de observações em 2D. Contudo, faz-se necessário tanto à triangulação quanto à recessão espacial utilizar-se dos parâmetros extrínsecos de imagens.

O processo de obtenção de parâmetros extrínsecos ou orientação exterior[6] requer a associação de coordenadas de imagens à pontos de controle. Pontos de

controle são pontos visíveis no espaço das imagens, cujas coordenadas do espaço tridimensional são conhecidas a priori. Esta associação permite ajustar um modelo matemático para determinar, segundo o referencial tridimensional adotado, a posição do centro de perspectiva (X_0, Y_0, Z_0) e ângulos de atitude (ω, ϕ, κ) do sistema de câmera no momento da obtenção da fotografia.

O conceito da condição de colinearidade da fotogrametria enuncia que, ao registrar uma foto, existe uma reta passando pelo centro de projeção perspectiva da câmera que liga cada ponto do objeto imageado a sua representação na imagem resultante. A figura 2.1 ilustra este conceito.

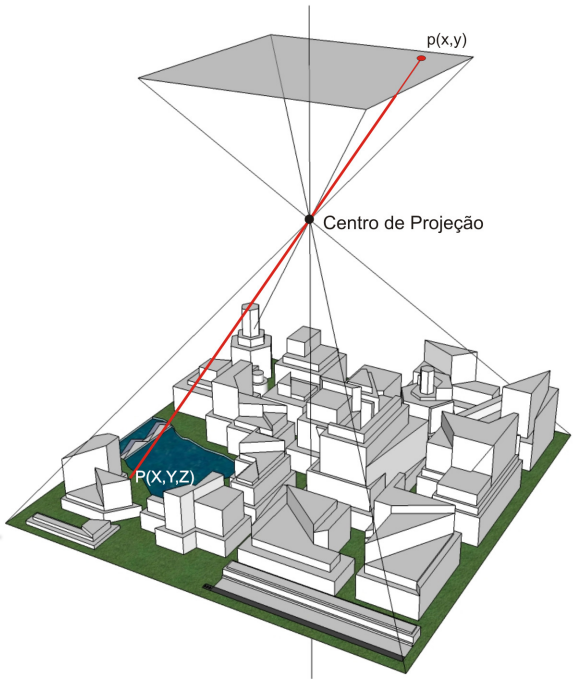

Figura 2.1: Condição de colinearidade.

Explorando este conceito é possível estabelecer equações de colinearidade para relacionar o centro de projeção $CP = (X_0, Y_0, Z0)$, um ponto genérico $P = (X, Y, Z)$ e a sua representação na imagem $p = (x, y)$:

$$\begin{cases} x = x_0 - f \times \dfrac{r_{11}(X - X_0) + r_{21}(Y - Y_0) + r_{31}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)} \\ y = y_0 - f \times \dfrac{r_{12}(X - X_0) + r_{22}(Y - Y_0) + r_{32}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)} \end{cases} \tag{2.2}$$

onde (x_0, y_0) refere-se ao centro óptico do sensor, f é a distância focal e os elementos $r_{i,j}$ pertencem a matriz de rotação R definida pelos ângulos de atitude do sensor ω, ϕ e κ em torno dos eixos X, Y e Z respectivamente.

$$R = R_\omega R_\phi R_\kappa \tag{2.3}$$

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\omega & sen\omega \\ 0 & -sen\omega & cos\omega \end{bmatrix} \begin{bmatrix} cos\phi & 0 & -sen\phi \\ 0 & 1 & 0 \\ sen\phi & 0 & cos\phi \end{bmatrix} \begin{bmatrix} cos\kappa & sen\kappa & 0 \\ -sen\kappa & cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Os métodos utilizados para obter a orientação exterior resolvem sistemas de equações de colinearidade de modo iterativo podendo tratar imagens individualmente, com técnicas de resseção espacial, ou em conjunto, usando a técnica de fototriangulação[6] que na língua inglesa é também conhecida pela nomenclatura *Bundle Adjstment*. Tais sistemas de equações podem exigir elevado poder computacional de modo que tais métodos também ocorrem na literatura científica associados a aceleração por processamento paralelo[26].

Alternativas para estes métodos incluem utilizar sensores adicionais na obtenção como o *Global Navigation Satellite System* (GNSS) e o *Inertial Navigation System* (INS). Com o uso de sensores adicionais o ajustamento matemático pode ser descartado, caso os sensores possuam precisão adequada, ou executado com inclusão das aproximações iniciais oriundas do GNSS e/ou INS para aumentar a aderência do modelo.

Figura 2.2: Fluxo de Trabalho proposto pela Estação Fotogramétrica Digital Livre do Projeto E-Foto (adaptado).

Um exemplo do fluxo de trabalho fotogramétrico pode ser verificado no diagrama apresentado pela figura 2.2 que demonstra a possibilidade de utilização de diferentes componentes para preparar as orientações de imagens. O destaque de tais componentes ocorre porque no contexto de uma estação fotogramétrica digital a orientação de imagens é requisito para outros processos além da extração de modelos digitais de elevações como, por exemplo, a restituição 3D[6] que cumpre o objetivo de viabilizar a interpretação vetorial das diferentes feições naturais ou artificiais que compõem uma região imageada.

2.1.2 Busca de Homólogos

Pontos homólogos caracterizam-se pela existência de pequenas porções de imagens distintas que registraram um mesmo ponto do espaço 3D. Sua localização automática pode ser realizada pelo uso do coeficiente de correlação de Pearson[6], também conhecido como Correlação Cruzada Normalizada (NCC - *Normalized Cross-Correlation*), um operador de correlação estatística largamente aplicado no processamento de imagens.

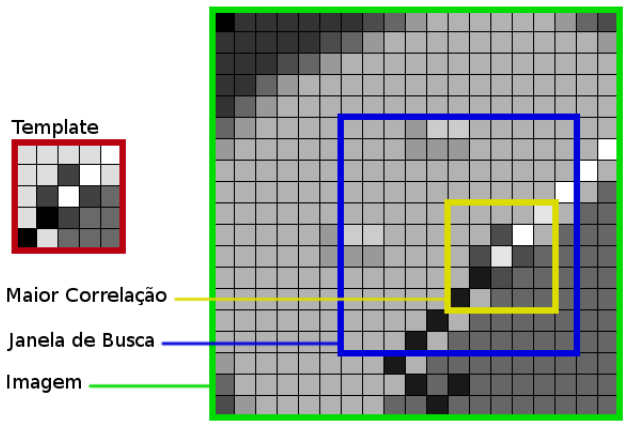

Figura 2.3: Conceitos de *template* (em vermelho), imagem (em verde), janela de busca (em azul) e amostra de maior correlação (em amarelo).

Segundo Stockburger[27] o cálculo de correlação pode ser realizado pela seguinte formulação:

$$\rho = \frac{\sigma I_1 I_2}{\sigma I_1 \sigma I_2} \tag{2.4}$$

onde:

$\sigma I_1 I_2$ é a covariância entre as duas variáveis I_1 e I_2;

σI_1 é o desvio padrão da variável I_1; e

σI_2 é o desvio padrão da variável I_2.

Os valores do coeficiente de correlação estão limitados entre -1 e 1. Um valor positivo indica o nível de correspondência a medida em que variam os valores das componentes de ambas as variáveis. Admite-se que zero equivale a nenhuma correlação e 1 refere-se a correlação total. Valores negativos indicam uma correlação negativa, ou seja, se um valor de uma componente aumenta numa variável ele diminui na outra e vice-versa.

Aplicada a imagens a técnica de NCC, figura 2.3, é normalmente enunciada pelo uso de um *template*, T, a ser comparado com uma imagem, I. O *template* é um pequeno pedaço de imagem e deve ser menor ou igual as dimensões da imagem à qual será comparado. Neste caso, a sua formulação pode ser ajustada para calcular os níveis de correlação para todos os possíveis posicionamentos de T sobre I:

$$\rho(k,l) = \frac{\frac{\sum_i \sum_j \left[I(i+k,j+l) - \overline{I}(i+k,j+l)\right]\left[T(i,j) - \overline{T}(i,j)\right]}{N-1}}{\sqrt{\frac{\sum_i \sum_j \left[I(i+k,j+l) - \overline{I}(i+k,j+l)\right]^2}{N-1}} \sqrt{\frac{\sum_i \sum_j \left[T(i,j) - \overline{T}(i,j)\right]^2}{N-1}}} \tag{2.5}$$

onde a notação $\overline{F}(x,y)$ refere-se a média da função F, ou seja, é $\frac{\sum_x \sum_y [F(x,y)]}{N}$ sendo N o número de *pixels* da amostra.

A busca de pontos homólogos consiste em determinar qual amostra da imagem possui maior nível de correlação ao ser comparada com o *template*, o que equivale a aplicar a função $\rho(k,l)$ e identificar onde ocorrem os máximos registrados para a função. Adicionalmente, um limiar de corte pode ser definido para eliminar máximos cujo índice de correlação seja insatisfatório aumentando assim a eficácia das buscas.

Contudo, o custo computacional para determinar todas as possíveis buscas de homólogos entre um par de imagens não é a alternativa viável se consideradas imagens de grandes dimensões. E ainda que fosse, poderia levar a falsos positivos quando uma região imageada possui alguma unidade estrutural que se repete ao longo da área registrada. Algumas melhorias tipicamente adotadas são:

- A parametrização de uma janela de busca, figura 2.3, reduzindo assim, a área de análise sobre a imagem;
- A inserção de pares de coordenadas, chamadas de sementes, em torno de onde estima-se que devem ocorrer pontos homólogos;
- A adoção de algoritmos para percorrer imagens aproveitando-se do conceito de vizinhança para realizar novas buscas de homólogos nas proximidades de pontos homólogos validados.
- A aceleração por GPU dos cálculos de correlações, como visto no trabalho de Arunagiri[28].

A alternativa comumente usada no processo fotogramétrico é eleger sementes reutilizando os pontos de controle usados para a produção dos parâmetros de orientação exterior. Uma vez que os processos de orientação exterior necessitam de poucos pontos, estes são tipicamente medidos ou validados por operadores humanos. Isto cumpre o papel de fornecer sementes para processos de busca fundamentados em análise das vizinhanças.

Ao explorar o conceito de vizinhança é possível adensar o número de pontos homólogos conhecidos. Isto é realizado por um algoritmo para crescimento de regiões que partindo de poucas sementes pode realizar buscas recursivas nas adjacências preenchendo assim a região de sobreposição das imagens sob análise.

A técnica de correlação cruzada normalizada é invariável a variações de contraste e brilho, mas encontra seus limites de aplicabilidade quando há grandes variações de escala e rotação entre as imagens. Neste sentido, técnicas de normalização de imagens[6] podem corrigir globalmente diferenças de escalas e rotações.

Outros métodos de correlação contemplando diversos descritores como o SIFT [29] são opções nos casos em que existem grandes variações no ângulo da tomada de imagens. Ele é aplicável no mapeamento topográfico, contudo as técnicas fotogramétricas antecedem a sua idealização e presumem condições de estabilidade de voo na tomada das fotografias aéreas de modo a minimizar as variações de atitude do sensor propiciando assim melhores condições para o uso da correlação de Pearson.

2.1.3 Triangulação

Na reconstrução de cenas o processo de triangulação, ilustrado pela figura 2.4, refere-se à determinação de pontos no espaço 3D dadas suas projeções em duas, ou mais, imagens. Tal processo está presente no contexto da literatura de fotogrametria, onde é também conhecido como interseção espacial[6].

Reorganizando as equações de colinearidade (2.2) para isolar as coordenadas (X, Y, Z) viabiliza-se a técnica. O que, para o caso de duas imagens, gera o sistema de equações, a seguir:

$$\begin{cases} X = X_{0_1} + (Z - Z_{0_1}) \times \alpha_{x_1} \\ Y = Y_{0_1} + (Z - Z_{0_1}) \times \alpha_{y_1} \\ X = X_{0_2} + (Z - Z_{0_2}) \times \alpha_{x_2} \\ Y = Y_{0_2} + (Z - Z_{0_2}) \times \alpha_{y_2} \\ Z = \dfrac{X_{0_2} - Z_{0_2}\alpha_{x_2} + Z_{0_1}\alpha_{x_2} - X_{0_1}}{\alpha_{x_1} - \alpha_{x_2}} \\ Z = \dfrac{Y_{0_2} - Z_{0_2}\alpha_{y_2} + Z_{0_1}\alpha_{y_2} - Y_{0_1}}{\alpha_{y_1} - \alpha_{y_2}} \end{cases} \tag{2.6}$$

onde:

$$\alpha_{x_i} = \frac{r_{11_i}(x_1 - x_{0_i}) + r_{12_i}(y_i - y_{0_i}) - r_{13_i}f}{r_{31_i}(x_i - x_{0_i}) + r_{32_i}(y_i - y_{0_i}) - r_{33_i}f}$$
$$\alpha_{y_i} = \frac{r_{21_i}(x_1 - x_{0_i}) + r_{22_i}(y_i - y_{0_i}) - r_{23_i}f}{r_{31_i}(x_i - x_{0_i}) + r_{32_i}(y_i - y_{0_i}) - r_{33_i}f} \tag{2.7}$$

No sistema de equações 2.6 há uma superabundância de dados e sua resolução pode ser dada pelo uso do método dos mínimos quadrados para um ou mais pares de imagens onde um mesmo ponto homólogo possa ser identificado.

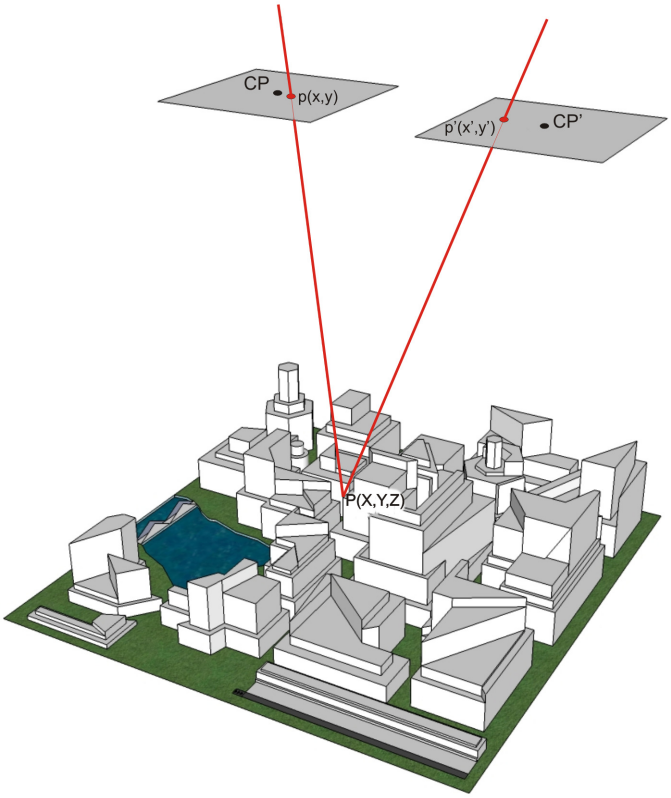

Figura 2.4: Triangulação de coordenadas.

2.2 Arquiteturas GPU

2.2.1 Visão Global

A área de computação de alto desempenho estuda entre outras questões a aplicação de máquinas de processamento paralelo para resolver problemas de elevado esforço computacional. Muitos destes problemas envolvem a aplicação repetitiva de operações, que representam um percentual pequeno do código conforme mostra a

figura 2.5, sobre seções de grandes massas de dados. Neste segmento placas de vídeo tornaram-se potenciais aceleradores de computação. Processadores embarcados em placas de vídeo modernas são chamados de *Graphics Proccessing Unit* (GPU) e são constituídos por múltiplos núcleos sendo fundamentalmente dispositivos para processamento de dados em paralelo[12].

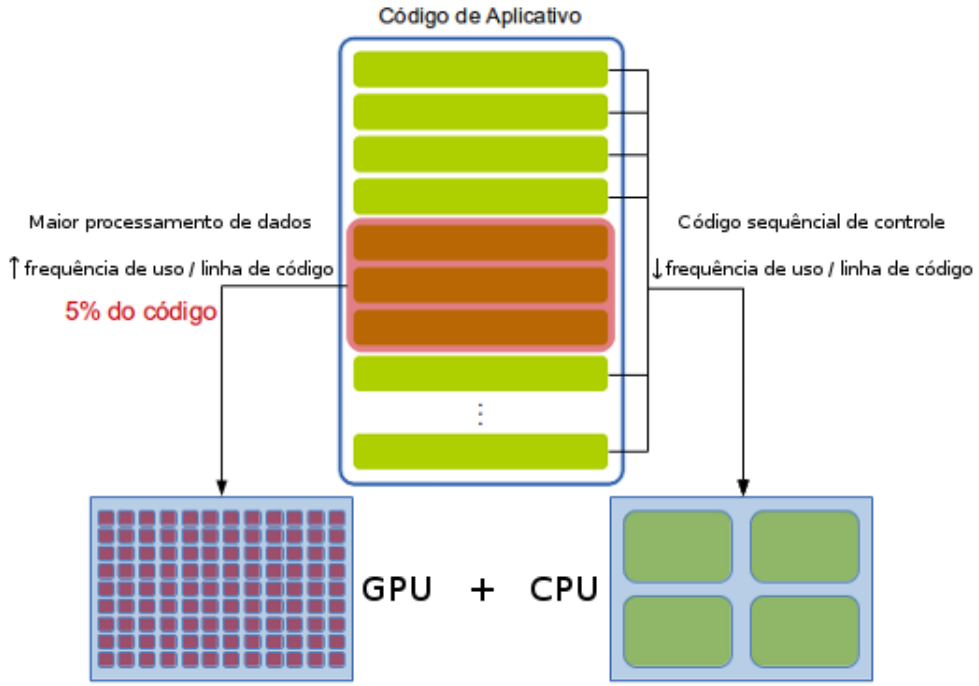

Figura 2.5: Como GPUs podem acelerar o processamento.

O termo *General-Purpose computation on Graphics Proccessing Unit* refere-se ao uso dos disponíveis nas GPUs para acelerar a solução de problemas de caráter geral. Isto equivale a estender o uso destes dispositivos para além da renderização gráfica, dando-os aplicações em, por exemplo, processamento de imagens, visão computacional, cálculo numérico, simulações físicas e etc. A difusão da terminologia conta com contribuições de Mark Harris, que fundou em 2002 um domínio[30] para registrar publicações científicas e demais esforços relacionados ao tema.

Um dos principais desafios nos anos iniciais de pesquisas para aceleração em GPU eram as linguagens associadas a estes dispositivos. Elas definiam um *pipeline* de processamento gráfico e primitivas próprias para aplicações de computação gráfica introduzindo dificuldades na reformulação das soluções de outros contextos. O que motivou as fabricantes a se adaptarem para responder a esta tendência generalizando os núcleos de processamento em suas arquiteturas.

No ano de 2006 ocorreu o lançamento da *Compute Unified Device Architecture* [19] (CUDA) para programação dos dispositivos de hardware da nVidia. No mesmo período foram publicados os primeiros sucessos científicos neste seguimento, onde ar-

quiteturas CPU foram superadas pelas arquiteturas GPU na resolução de operações matriciais. Tais feitos impulsionaram iniciativas de outros fabricantes como a Microsoft com o DirectCompute, a AMD com o FireStream e o openCV, uma iniciativa de padrões abertos, tal como, o openGL e o openCV.

A plataforma para computação paralela, CUDA, simplificou a exploração do potencial das GPU pela integração com aplicações desenvolvidas em linguagens C, C++ e FORTRAN. Seu compilador especializado, o *nvcc*, abstrai a complexidade intrínseca da montagem de código objeto para arquiteturas heterogêneas e funciona como uma extensão do compilador nativo, *gcc* em Linux ou *MVS compiler* em Windows, encaminhando tarefas a estes compiladores e suportando algumas opções de linha de comando adicionais próprias à arquitetura das GPUs.

O ambiente de desenvolvimento de aplicações para GPU é comumente composto com um dispositivo de hardware capacitado, o *driver* apropriado, o compilador nativo do sistema operacional e o *toolkit*. As primeiras placas a terem núcleos capacitados em CUDA foram as placas da família GeForce 8k possuindo entre 8 e 96 *cores*. Atualmente são muitos os dispositivo nVidia que possuem núcleos CUDA tendo chegado ao número de 5760 núcleos no lançamento mais recente[31].

2.2.2 Conceitos

Para permitir o processamento de grandes quantidades de dados de forma escalável os processos em execução na GPU, chamados de *threads*, são divididos em blocos e, por sua vez, ao conjunto de blocos se dá o nome de *grid*. A distribuição de processos é ilustrada pela figura 2.6. Um *grid* pode ser estruturado em uma, duas ou três dimensões e durante o seu processamento são gerados índices para permitir a localização de seus blocos e *threads*. *Threads* de um mesmo bloco dispõem de uma memória compartilhada e pode haver comunicação entre eles, contudo os blocos são independentes.

Os núcleos na GPU são distribuídos em unidades de processamento chamadas de *streaming multiprocessor*. Normalmente o número de núcleos disponível é um múltiplo de 8, menor ou igual a 192, dependendo da versão da arquitetura. Cada bloco em execução é subdividido em *warps* para permitir a alocação de *threads* em cada um dos núcleos disponíveis no *multiprocessor*. Todas as *threads* num *warp* executam instruções simultaneamente e quando há ociosidade devido à latência de memória ou pela necessidade de sincronização ocorre a troca de contexto do *warp* ativo. Esta troca pode ser realizada rapidamente e assim um *multiprocessor* pode se ocupar do controle de execução de um elevado número *threads*.

Processos desenvolvidos para executar num dispositivo CUDA são descritos como funções em linguagem C admitindo-se comandos adicionais previstos para viabilizar

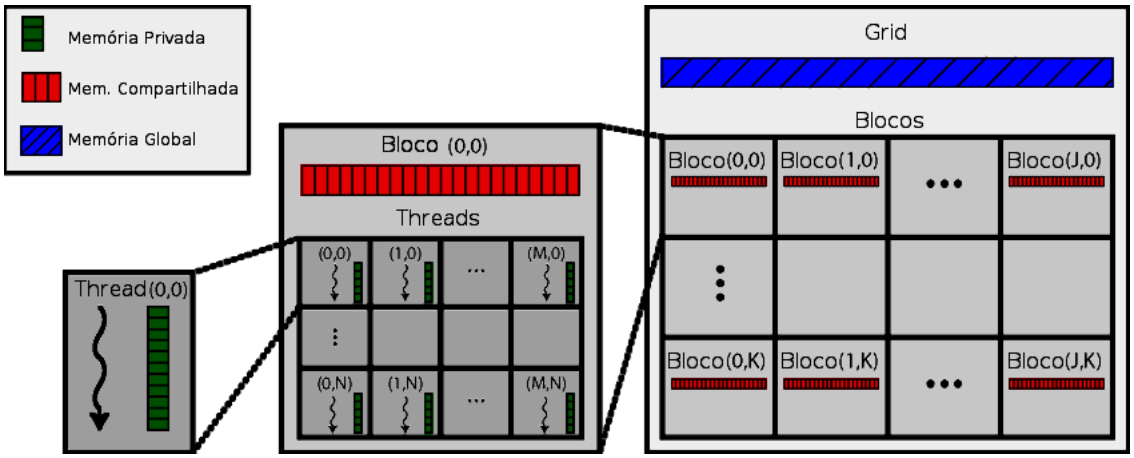

Figura 2.6: Modelo de distribuição de processos para execução em GPU.

a computação paralela. Estes processos são chamados de *kernel* e durante sua invocação devem ser passados, além dos parâmetros para execução, as configurações de controle como, por exemplo, dimensões dos blocos usados e número de *threads*. Um *kernel* pode ser desenvolvido para responder a chamados da CPU ou de outros processos executando no dispositivo GPU, contudo não admite-se o uso de chamadas recursivas.

Durante a construção de um *kernel* devem ser consideradas os diversos tipos de memória previstos pela hierarquia de memória. Normalmente é feita uma cópia entre a memória da CPU e a memória global de GPU para disponibilizar dados para um *kernel*. A memória global possui muito espaço, mas com alta latência se comparada com outras opções de memória. Uma fração da memória cache que opera junto da memória global pode ser definida como constante, reduzindo a latência para acesso dos dados carregados antes da execução do *kernel*, contudo isto se limita às dimensões da *cache*. Outra opção de memória global é a memória de texturas e seu uso pode ser otimizado para aproveitamento da localidade espacial durante o acesso de seu conteúdo em uma, duas ou três dimensões. A opção de memória de menor latência é a memória compartilhada, contudo ela é pequena e limitada ao escopo de um bloco. Os núcleos dispõem de alguns registradores que implementam a memória privada para manter as variáveis internas de cada *thread*. Quando estes registradores não são suficientes o excedente é alocado na memória compartilhada do bloco. Por fim, em casos onde a massa de dados é muito grande para a memória da GPU há opções para paginação de memória da CPU.

Dentre os padrões de programação adotados no desenvolvimento de soluções paralelizáveis encontra-se a classe de algoritmos de redução. Algoritmos de redução[19] caracterizam-se por entradas de múltiplos elementos, $O(N)$ no caso mais simples, e tratam da aplicação de um mesmo operador associativo ao conjunto de elementos

resultando num único elemento, $O(1)$, em sua resposta. Os operadores associativos comuns são soma, soma de quadrados, produto, máximo, mínimo ou operadores binários *AND* e *OR*. Especificamente neste trabalho há grande ocorrência de somatórios para computar cada índice de correlação nas buscas de homólogos e também o uso de operadores de máximo ao definir a máxima correlação de cada busca.

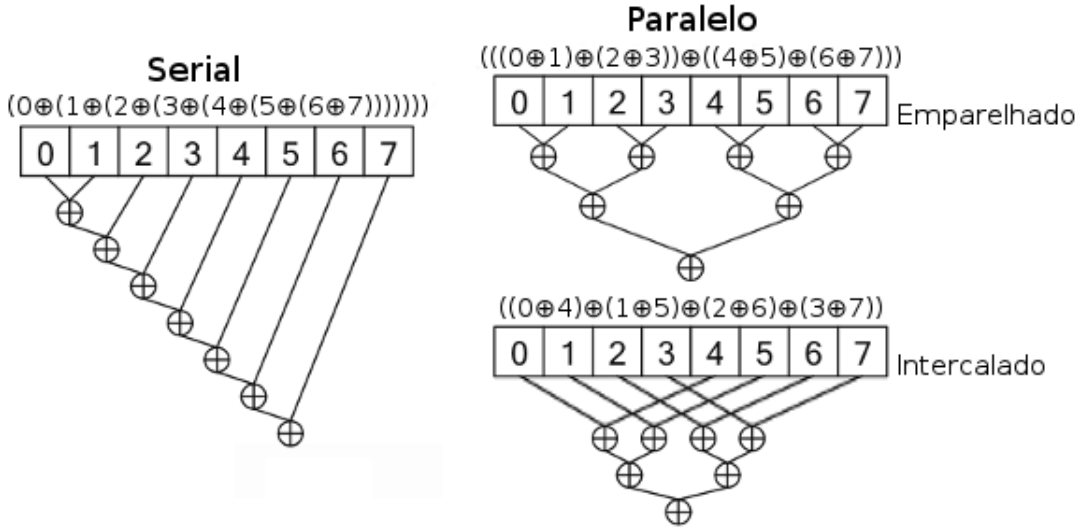

Figura 2.7: Redução de 8 elementos (adaptado de CUDA Handbook).

Enquanto algoritmos seriais em geral utilizam-se de N - 1 ciclos para efetuar a redução, os algoritmos paralelos dividem o esforço entre diversos processos podendo atingir a solução em $O(lgN)$ ciclos. Isto é demonstrado na figura 2.7, onde é ilustrado um vetor de 8 elementos para o qual a solução serial é atingida em 7 passos e as soluções paralelas ocorrem em 3 passos. O acesso a dados normalmente pode ser feito em pares emparelhados ou em forma intercalada, contudo a implementação de redução em paralelo requer cuidados especiais para evitar ociosidades devido ao tempo de leitura de memória e conflitos de escrita. Assim, soluções em CUDA tendem a adotar a computação de resultados parciais por blocos emparelhando-os em relação aos dados de modo a melhor aproveitar a localidade de memória. Tipicamente são usados acessos intercalados dentro dos blocos para facilitar o endereçamento e reduzir a divergência de *threads* por *warp*.

Capítulo 3

Método Proposto

Este capítulo apresenta a sequência de tarefas realizadas, bem como os recursos envolvidos no desenvolvimento da solução proposta para o tema de extração de modelos digitais de elevações acelerada em GPU. A metodologia utilizada define quatro etapas, a saber: análise do problema, desenho da solução, implementação e testes. Enquanto as primeiras três etapas são abordadas respectivamente nos itens 3.1, 3.2 e 3.3 deste capítulo a parte de testes se reserva ao capítulo 4.

3.1 Algoritmo Sequencial

A proposta deste trabalho partiu de uma implementação preexistente, de extração do modelo digital de elevações para pares de imagens estereoscópicas, com o objetivo de aceleração e por este motivo fez-se necessário colher parâmetros de sua execução como referência.

Suas características construtivas permitem processamento das imagens disponíveis no projeto fotogramétrico na escala original ou em escalas reduzidas. A identificação de uma densa quantidade de pontos homólogos sobre a área de sobreposição das imagens é possível pela adoção de uma estratégia de crescimento de regiões. A construção do crescimento de regiões foi feita pela adaptação da vertente não recursiva do algoritmo de preenchimento, *floodfill*, para 4 direções e usando pilhas como estrutura de armazenamento para as iterações. Uma vez que o crescimento de regiões identifique os pontos homólogos é realizada a triangulação de pontos em 3D constituindo-se o modelo.

Os trechos, a seguir, em pseudocódigo foram descritos para demonstrar de modo resumido a sequência de operações envolvidas no processo de modo a melhorar o entendimento durante a análise do mesmo que será apresentada na seção 3.1.1, mais a frente.

```
Extração:
{
  Aplique a escala adotada às sementes;
  Para cada par de imagem faça a extração do par:
  {
    Carregue as imagens em memória e prepare para a escala adotada;
    Para cada semente faça o crescimento de regiões:
    {
      Coloque a semente na pilha de regiões a processar;
      Enquanto houverem regiões a processar faça a iteração de
         crescimento:
      {
        Retire uma semente da pilha de regiões a processar;
        Se região não foi visitada e não está fora dos limites das
           imagens então:
        {
          Marque a região como visitada;
          Faça a busca de homólogos com a semente:
          {
            Para cada possível posição do template sobre a janela
               de busca faça a verificação do índice de correlação:
            {
              Execute a função de correlação;
              Se correlação é maior que a anterior então:
              {
                Armazene valor e posição do máximo de correlação;
              }
            }
          }
          Se o resultado da busca supera o limiar de corte então:
          {
            Ajuste a semente conforme o posicionamento do máximo;
            Armazene o resultado;
            Coloque na pilha a semente para o vizinho da esquerda;
            Coloque na pilha a semente para o vizinho da direita;
            Coloque na pilha a semente para o vizinho abaixo;
            Coloque na pilha a semente para o vizinho acima;
          }
        }
      }
    }
  }
  Redimensione os resultados considerando a escala adotada;
  Triangule as coordenadas 3D;
}
```

O código fonte do programa preexistente foi desenvolvido segundo o paradigma de orientação a objetos e está descrito em linguagem C++. Os métodos que implementam este algoritmo são pertencentes em sua grande maioria a um conjunto de 5 classes, a saber:

- Classe controladora de extração do modelo digital de elevações, que gerencia a extração e mantém atualizada a interface gráfica;
- Classe de interface gráfica, que durante a extração fornece parâmetros de configuração, além de embarcar código para a abertura das imagens;
- Classe de processamento de imagens, que realiza o crescimento de regiões e comanda as buscas de correlação;
- Classe de correlação cruzada normalizada, que implementa os cálculos necessários à correlação;
- Classe de interseção espacial, que implementa os cálculos necessários ao processo de triangulação de pontos 3D.

A divisão da sequência de execução com estas classes pode ser vista na figura 3.1, que apresenta o diagrama de sequência simplificado para o caso geral de execução, onde se destacam as estruturas de repetição, mas encontram-se omitindo rotas alternativas que contemplam o tratamento de exceções. Sobre o diagrama são ressaltados pontos quentes para o objetivo de otimização, onde são realizadas somas necessárias ao cálculo dos índices de correlação.

Durante a busca de homólogos são executadas mxm correlações, onde m é a dimensão da janela de busca. O cálculo da correlação é segmentado em diferentes rotinas para computar as funções média, desvio padrão e covariância. Em cada uma destas rotinas são acessadas nxn posições de memória nas duas imagens sob análise, onde n é a dimensão do *template*. Por consequência deste desmembramento da correlação em diferentes rotinas ocorre um agravamento do esforço computacional levando a releitura das mesmas posições de memória diversas vezes seguidas.

Para gerar maior compreensão do problema, explicitar dados sobre o custo associado a sua execução e maximizar os esforços de otimização no núcleo de maior consumo dos recursos computacionais, foram realizadas medições e análises com as ferramentas, *Gcov* e *Gprof*, distribuídas em conjunto do compiladores de linguagem C e C++ disponíveis em ambientes linux, o *gcc* e o *g++*.

Tendo em vista o uso intensivo do cálculo do índice de correlação e a consequente quantidade elevada de somatórios realizados a estimativa inicial apontava para a necessidade de otimizar a operação de busca de homólogos e reduzir o número de acessos a memória. Contudo o uso das ferramentas de análise neste estudo contribuíram para definir outros meios de aceleração incorporados ao escopo de trabalho.

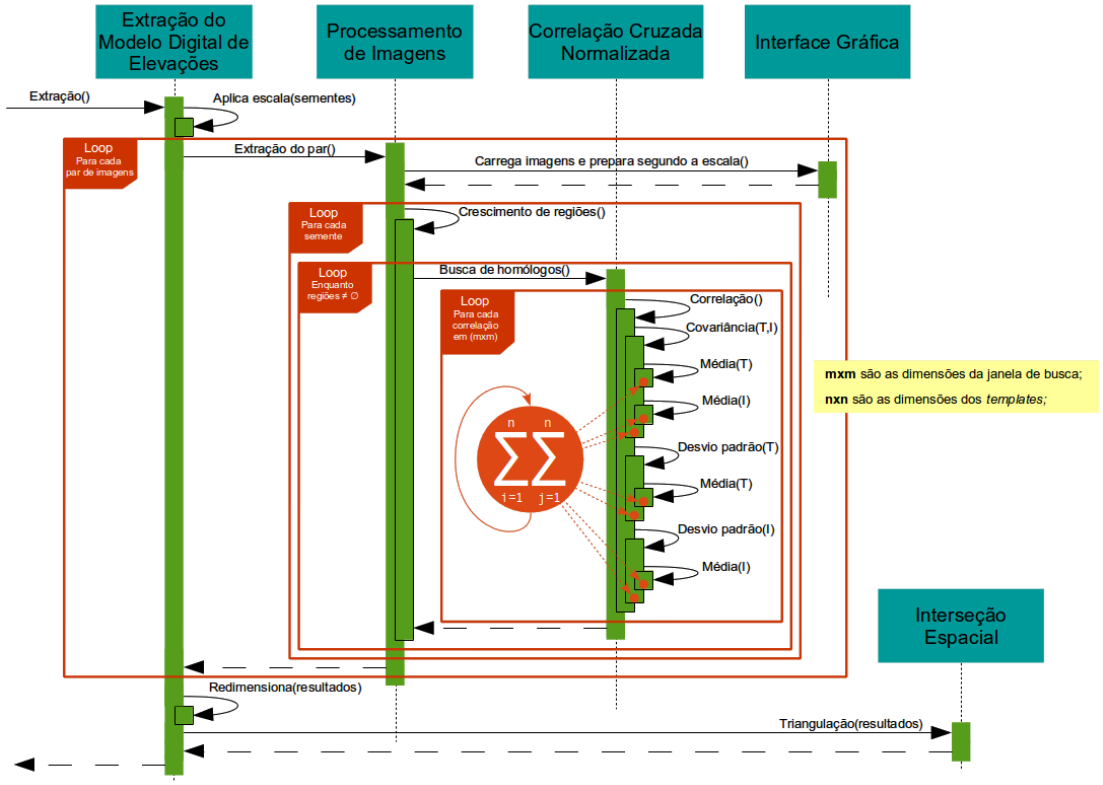

Figura 3.1: Diagrama de sequência simplificado para o programa preexistente. Destacam-se pontos quentes para a otimização, onde ocorrem somatórios.

3.1.1 Análise com *Gprof* e *Gcov*

O método de análise pelo uso das ferramentas empregadas foca na criação de códigos mais eficientes reduzindo o número de falhas, viabilizando ajustes em rotinas de alto custo e por consequência aumentando sua velocidade de execução final. Neste sentido, o papel do *Gprof* foi coletar e organizar os dados para análise no nível de rotinas, ao tempo que o *Gcov* foi contador no nível de linha de código permitindo assim uma análise mais detalhada no conteúdo das rotinas de maior interesse.

A integração dos recursos oferecidos por estas ferramentas implicou na recompilação dos códigos fonte com a adição de diretivas necessárias ao seu funcionamento. Tais diretivas foram anexadas ao arquivo de projeto definido pelo *framework* Qt para automatização da escrita do *Makefile*[1].

O uso do *Gprof* objetiva a criação de perfis de dados sobre as funções[2] chamadas durante o tempo de execução do programa. Em tempo de execução o programa recompilado grava um arquivo de *log* chamado *gmon.out*. Deste arquivo foram extraídos resultados em dois formatos principais, a saber:

[1] *Makefile* é o arquivo que organiza a compilação.

[2] Vale lembrar que os métodos do C++, no contexto de orientação a objetos, são tratados como funções pelo compilador.

- *Flat Profile* - uma tabela das funções executadas segundo o tempo despendido e o número de chamadas. A tabela 3.1 resume o resultado, onde observa-se que um conjunto de quatro funções (método de acesso, desvio padrão, média e covariância) concentram mais de 96% de todo o tempo de execução;

Tabela 3.1: Resumo do relatório, *flat profile*, criado pelo *Gprof* para demonstrar o perfil do tempo de execução para o programa preexistente.

Tempo (%)	(%) Acum.	Tempo Próprio	Número de Chamadas	Função
64,18	64,18	48,00	10.979.743.212	Método de acesso
12,60	76,78	9,42	56.422.215	Desvio padrão
11,85	88,63	8,86	112.795.965	Média
7,78	96,41	5,82	28.186.875	Covariância
0,62	97,03	0,46	28.186.875	Correlação
0,39	97,42	0,29	45.099	Busca de homólogos
0,17	97,59	0,13	4	Carregamento de imagem
0,11	97,70	0,08	975.364	Alocação de memória
0,08	97,78	0,06	769.994	Cópia de memória
0,05	97,83	0,04	11.024.241	Escrita em memória
0,03	97,86	0,02	10.681.311	Contagem de colunas
0,01	97,84	0,01	1.232.092	Limpeza de memória
0,01	97,85	0,01	102.658	Multiplicação de matrizes
0,01	97,86	0,01	12	Crescimento de regiões

- *Call Graph* - um grafo de chamadas mapeando como as funções estão encadeadas. Neste resultado são expressos percentuais do tempo de execução de um nó considerando seus dependentes, o percentual de tempo próprio de execução para cada função e o número de vezes que a função foi chamada. A figura 3.2 demonstra a ramificação derivada da função de crescimento de regiões.

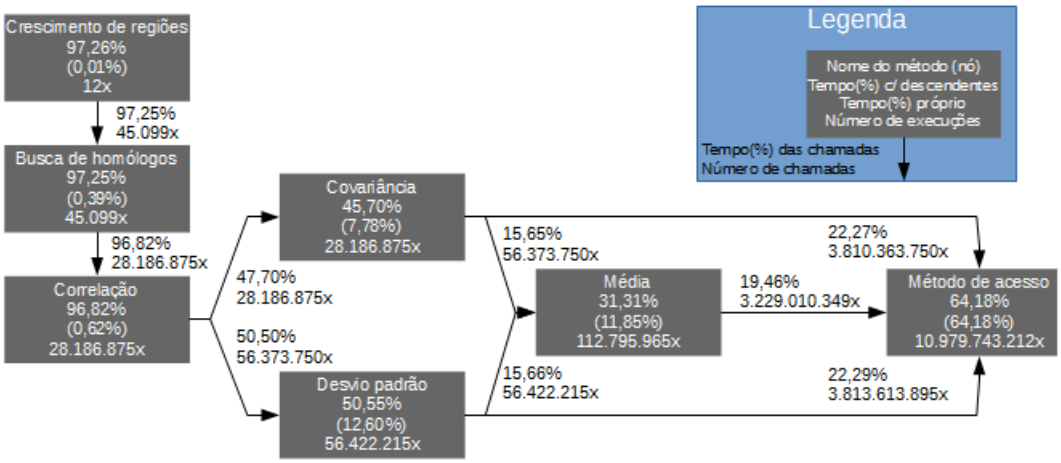

Figura 3.2: Visão da ramificação que despende maior tempo de execução no programa preexistente. Adaptada do relatório, *call graph*, gerado com o *Gprof*.

Analisando o conjunto de dados sobre o tempo de execução do programa sequencial preexistente foi possível identificar quais partes do programa necessitavam de maior atenção de otimização. O cruzamento dos valores percentuais de tempo de execução e dos números de chamadas para cada função pôde ser usado para estimar o tempo médio de execução por chamada, além de apontar funções com elevado grau de chamadas. Isto foi fundamental para evidenciar uma possível grande perda de desempenho devido ao encapsulamento de valores na classe de cálculos matriciais usada pelo programa. Conforme mostra a tabela 3.1 mais da metade do tempo gasto pelo programa se devia ao método de acesso.

Tal visão foi complementada pela avaliação dos resultados do *Gcov* tendo em vista que sua utilização permite contar, em tempo de execução, o número de vezes que cada linha do código fonte é usada. Isto permite segmentar o código segundo a frequência de uso de suas diferentes partes.

Um exemplo dos resultados obtidos pode ser visualizado na figura 3.3, onde podem ser identificadas, numa escala logarítmica, duas das linhas do código com maior frequência de uso, sendo estas as linhas do método de acesso (*get*) da classe de cálculos matriciais (*Matrix*) que executam validação de endereçamento e retorno do valor armazenado em memória.

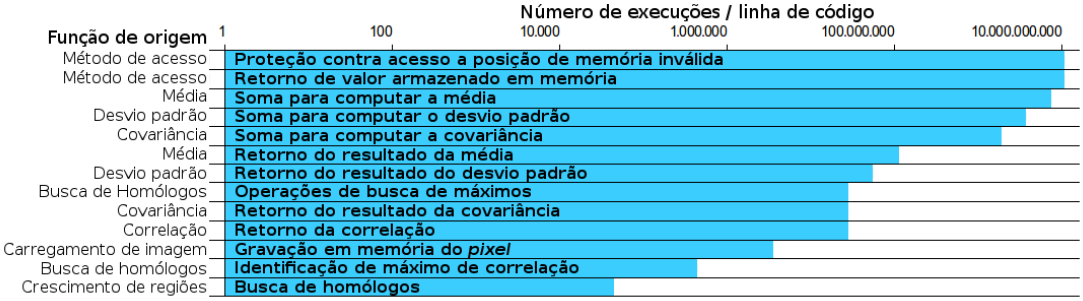

Figura 3.3: Gráfico de frequência do uso em escala logarítmica para algumas das linhas de código do programa preexistente. O valores foram observados após a execução do programa integrado com o *Gcov*.

Embora o tempo da troca de contexto para execução de funções no processador possa ser muito pequeno, se uma função é chamada muitas vezes a soma de todas as chamadas pode representar uma fatia consideravelmente elevada no tempo de execução do programa. Deste modo, o uso excessivo do método *get* indica uma possibilidade de aceleração simplificando o código para que a busca de homólogos possa fazer acessos diretos a memória. Isto fundamenta uma iniciativa de alteração da estrutura usada, no caso a classe *Matrix*, para armazenar imagens durante a avaliação de correlações.

A etapa de análise foi finalizada com a realização de uma modificação no código

preexistente dando acesso direto ao processo de busca de homólogos para os valores armazenados pela classe matricial de modo a demonstrar qual ganho em desempenho seria aplicável.

Isso foi realizado com a substituição das chamadas ao método *get* no código fonte por macros. Além disso um dos atributos da classe *Matrix*, referente ao ponteiro para os elementos da matriz alocados em memória, teve sua regra de acesso modificada para garantir o uso público. Por fim, o algoritmo foi revisado no nível da chamada de uma correlação de modo a garantir proteção a falhas que levariam a acessos a posições de memória inválidas. Tais modificações são ilustradas a seguir:

```
#ifdef WITH_DIRECT_ACCESS
#define MATRIXGET(M,i,j) (M._Mat[(int)(((i)-1)*M.ncols+((j)-1))])
#define MATRIXPGET(M,i,j) (M->_Mat[(int)(((i)-1)*M->ncols+((j)-1))])
#endif
#ifndef WITH_DIRECT_ACCESS
#define MATRIXGET(M,i,j) ( M.get(i,j))
#define MATRIXPGET(M,i,j) ( M->get(i,j) )
#endif

class Matrix {
    (...)
public:
    double* _Mat;
    (...)
};

double NCC::correlation(...)
{
    (...)
    if (row_I1_i < 0 || col_I1_i < 0 ||
        row_I2_i < 0 || col_I2_i < 0 ||
        row_I1_f > I1->getRows() || col_I1_f > I1->getCols() ||
        row_I2_f > I2->getRows() || col_I2_f > I2->getCols())
            return 0;
    (...)
}

double NCC::covXY(...)
{
    (...)
    sum += (MATRIXPGET(X,i,j) - avgX)
           *(MATRIXPGET(Y,i+delta_i, j+delta_j) - avgY);
    (...)
}
```

```
double NCC::stddev(...)
{
    (...)
    sum += pow(MATRIXPGET(m, i, j) - avg, 2);
    (...)
}

double NCC::average(...)
{
    (...)
    sum += MATRIXPGET(m, i, j);
    (...)
}
```

Os resultados associados modificação no código preexistente serão apresentados e discutidos no capítulo 4. Contudo, vale ressaltar que embora o método de acesso tenha sofrido modificações, o número total de acessos a memória não foi modificado. Assim é evidente a possibilidade de maiores esforços de otimização reduzindo a segmentação da rotina de correlação e por consequência aumentando o desempenho do código tendo em vista que as rotinas associadas ao cálculo das correlações ocupam uma maior parte de tempo durante a execução do programa.

3.2 Otimização

Após serem realizados os testes com a modificação no código do programa preexistente, conforme cita o item anterior, iniciou-se a etapa de desenvolvimento da solução proposta. Ela reaproveita objetos fotogramétricos implementados no programa preexistente, contudo redefine o controle da extração e as estruturas de dados usadas de modo a aumentar o desempenho.

A estratégia principal da otimização foi fundamentada em resposta à grande quantidade de somatórios utilizados nos processo de extração. Foram verificadas as possíveis melhorias para a implementação da equação do índice de correlação.

A equação (2.5) pôde ser reescrita com a remoção dos denominadores comuns (N - 1).

$$\rho(k,l) = \frac{\sum_i \sum_j \left[I(i+k,j+l) - \overline{I}(i+k,j+l)\right]\left[T(i,j) - \overline{T}(i,j)\right]}{\sqrt{\sum_i \sum_j \left[I(i+k,j+l) - \overline{I}(i+k,j+l)\right]^2 \sum_i \sum_j \left[T(i,j) - \overline{T}(i,j)\right]^2}} \quad (3.1)$$

Indo além, segundo sugeriu Wilt[19], esta formulação pôde ainda ser reescrita substituindo as médias $\overline{I}$ e $\overline{T}$ em função do reaproveitamento dos somatórios.

Sem a notação de coordenadas assumiu-se a seguinte formulação:

$$\rho = \frac{N\Sigma IT - \Sigma I \Sigma T}{\sqrt{[N\Sigma I^2 - (\Sigma I)^2]\,[N\Sigma T^2 - (\Sigma T)^2]}} \tag{3.2}$$

onde os somatórios que seriam realizados repetidas vezes para computar os valores de média, covariância e desvio padrão podem ser computados simultaneamente reduzindo o número de instruções ao executar as correlações.

Isto permitiu a definição de variáveis, conforme mostra a tabela 3.2, usadas para substituir a estrutura de dados anterior durante o desenvolvimento da versão de código paralelo. A implementação para esta formulação é demonstrada a seguir:

```
inline float correlation_o1( float SumI, float SumISq,
                             float SumIT, float SumT,
                             float SumTSq, float N)
{
    float Numerator = N*SumIT - SumI*SumT;
    float Denominator = sqrtf((N*SumISq - SumI*SumI)
                             *(N*SumTSq - SumT*SumT));
    return Numerator / Denominator;
}
```

Tabela 3.2: Variáveis adotadas no código otimizado.

Nome	Valores
SumI	ΣI
SumISq	ΣI^2
SumT	ΣT
SumTSq	ΣT^2
SumIT	ΣIT

A vantagem desta função de correlação é a externalização dos somatórios possibilitando a variação dos métodos de computação das somas. Demais melhorias podem ser alcançadas se observado que numa busca de homólogos um mesmo *template* é comparado com as diversas amostras presentes na janela de busca. Isto permite reduzir o número de operações pela pré-computação do termo $N\Sigma T^2 - (\Sigma T)^2$. A implementação pôde ainda ser melhorada substituindo-se a operação de divisão com o uso da função de cálculo do recíproco da raiz que apresenta maior eficiência. A seguir é demonstrada a versão do código sequencial contemplando estas melhorias para executar as diversas correlações associadas a uma busca e produzir o resultado ajustado.

```
#define GET(I, cols, i, j) (I[(i) * (cols) + (j)])
#define f(value) (float(value))

inline float correlation_o2( float SumI, float SumISq,
                             float SumIT, float SumT,
                             float N, float DenomPart)
{
    float Numerator = N*SumIT - SumI*SumT;
    float Denominator = rsqrtf((N*SumISq - SumI*SumI)*DenomPart);
    return Numerator * Denominator;
}

// Busca de um par de homólogos dada uma semente
homologousAdjustment search_homologous(unsigned char *I1,
                                       unsigned char *I2,
                                       int cols_I1, int cols_I2,
                                       controlParameters cp,
                                       homologousAdjustment seed)
{
    unsigned int SumI, SumISq, SumT, SumTSq, SumIT;
    int dx, dy;
    float value, max_P = -1.0f, N = float(cp.tmpltDim * cp.tmpltDim);
    int tmpltX = seed.tmplt_x, tmpltY = seed.tmplt_y;
    int imageX = seed.image_x, imageY = seed.image_y;
    homologousAdjustment result = seed;

    // Calcula a parte constante do denominador
    SumT = SumTSq = 0;
    for (int y = 0; y < cp.tmpltDim; y++)
    {
        for (int x = 0; x < cp.tmpltDim; x++)
        {
            unsigned char T = GET(I1, cols_I1,
                                  tmpltY+cp.tmpltOffset+y,
                                  tmpltX+cp.tmpltOffset+x);
            SumT += T;
            SumTSq += T*T;
        }
    }
    float DenomPart = f(N*SumTSq) - f(SumT*SumT);

    // Calcula correlações dada uma semente
    for (int i = 0; i < cp.roiDim; i++)
    {
        for (int j = 0; j < cp.roiDim; j++)
        {
```

```
            SumI = SumISq = SumT = SumIT = 0;
            for (int y = 0; y < cp.tmpltDim; y++)
            {
                for (int x = 0; x < cp.tmpltDim; x++)
                {
                    unsigned char T = GET(I1, cols_I1,
                                          tmpltY+cp.tmpltOffset+y,
                                          tmpltX+cp.tmpltOffset+x);
                    unsigned char I = GET(I2, cols_I2,
                                          imageY+cp.roiOffset+i+y,
                                          imageX+cp.roiOffset+j+x);
                    SumT += T;
                    SumI += I;
                    SumISq += I*I;
                    SumIT += I*T;
                }
            }
            value = correlation_o2( f(SumI), f(SumISq),
                                    f(SumIT), f(SumT),
                                    N, DenomPart );
            if (value > max_P)
            {
                max_P = value;
                dx = j;
                dy = i;
            }
        }
    }

    // Ajusta o resultado
    if (max_P > cp.threshold)
    {
        result.adjusted = true;
        result.corrMax = max_P;
        result.image_x += cp.roiOffset + dx;
        result.image_y += cp.roiOffset + dy;
    }
    else
        result.adjusted = false;

    return result;
}
```

3.3 Algoritmo Paralelo

A solução proposta neste trabalho foi compilada com o nome *pdem*, acrônimo para *Parallel Digital Elevation Model*, e foi escrita para executar a extração do modelo digital de elevações com as otimizações da busca de homólogos descrita anteriormente. Tal como no programa preexistente o nível de otimização utilizado na compilação foi o máximo, ou seja, foi passada para o compilador a diretiva *-O3*.

Foram definidos dois modos de execução para a solução proposta com objetivo de permitir a execução serial ou em paralelo. Sua execução se dá por linha de comando com a seguinte sintaxe:

pdem *<**filename**>* [***options***]

onde *filename* é um arquivo EPP e as opções são:

- -m MODE, onde MODE pode ser *parallel (default)* ou *serial*;
- -c CONF, onde CONF é o caminho para um arquivo de configuração;

O formato de arquivo EPP, de *e-foto photogrammetric project*, é um arquivo XML (*eXtensible Markup Language*) contendo dados de projetos desenvolvidos com a estação fotogramétrica digital, tais quais, imagens, pontos de controle, informações de calibração do sensor e etc. Foram definidos novos elementos segundo o formato de arquivo XML para parametrizar a extração. Tais parâmetros podem ser embarcados no arquivo EPP ou alternativamente carregados a parte com uso da opção de apontamento para um arquivo de configuração.

Os parâmetros configuráveis são:

border Valor de afastamento das bordas ($default = windowDim/2$);

downsample Escala para redução (aplica-se $1/downsample$ às imagens);

pairs Número do par de imagens que será extraído (de 1 ao total de pares; pode ser usado 0 para todos);

step Passo do crescimento de regiões em *pixels* (maior que 3);

threshold Limiar de corte (de 0,5 a 1);

tmpltDim Tamanho do *template* (de 5 a 50);

windowDim Tamanho da janela de busca (de 7 a 150);

Um ganho que pode ser explorado pela solução paralela deve-se à possibilidade de executar simultaneamente as diversas correlações de uma mesma busca

de homólogos. E devido a possibilidade de uso do programa com pequenas dimensões para as janelas de busca e/ou para *templates*, de modo a adequar o volume de dados disponibilizados a cada chamada dos processos da GPU, a implementação foi idealizada para acumular diversas buscas necessárias ao crescimento de regiões numa mesma invocação do *kernel*. Deste modo o programa paralelo prevê, além dos ganhos numa busca de homólogos individual, à possibilidade de executar diferentes buscas de homólogos simultaneamente.

3.3.1 Divisão do Processamento

Diferente do processo sequencial, na versão paralela, é importante caracterizar quais são as tarefas da busca de homólogos e planejar uma redistribuição destas considerada a possibilidade de colisões na escrita durante o ajustamento dos pontos homólogos. A redistribuição do fluxo também visa evitar a perda de desempenho quando consideradas as instruções condicionais características da avaliação do máximo de correlação que podem acarretar em divergências de *threads* num *warp*. A versão paralela deve então assumir que todas as correlações foram determinadas antes de iniciar a identificação dos máximos e ajustamentos.

A figura 3.4 ilustra as etapas do programa implementado neste trabalho. Em nível macro o programa lê dados de projeto e configuração, processa a extração e grava resultados. A extração propriamente pode ser dividida em preparação das imagens, crescimento de regiões e triangulação de pontos.

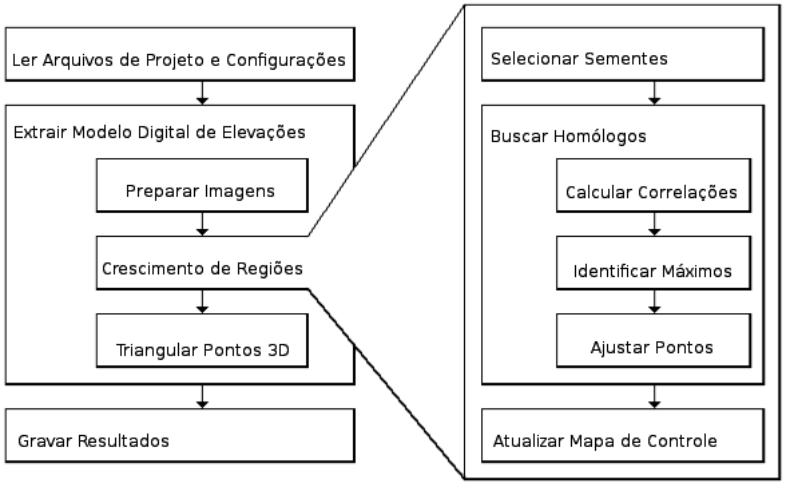

Figura 3.4: Fluxo geral de execução.

As etapas de preparação de imagens e triangulação de pontos 3D são processadas sequencialmente tendo em vista que suas representatividades no tempo total de execução é baixa. Enquanto duas funções para a etapa de crescimento de regiões foram projetadas com o intuito de atender aos modos de execução serial e paralelo.

A etapa de preparação inclui o ajustamento radiométrico de modo a aproximar o histograma das imagens em processamento e a cópia destas para a GPU. De modo a evitar um elevado número de cópias de recortes das imagens associados a cada *template* e janela de busca, com a percepção de que muitos destes recortes possuiriam sobreposição nas fronteiras de buscas, foi adotado a carga como texturas do par de imagens em estudo na memória da GPU. Assim, durante as iterações necessárias a execução do crescimento de regiões, as cópias entre a memória da CPU e da GPU limitam-se a gravação das sementes e dos resultados de cada busca de homólogos.

A cópia das imagens é feita com o seguinte trecho de código:

```
cudaChannelFormatDesc desc;
texture<unsigned char, 2> texImage;
texture<unsigned char, 2> texTmplt;
cudaArray *pArrayImage = NULL;
cudaArray *pArrayTmplt = NULL;

// Alocando espaço no dispositivo
CUDART_CHECK( cudaMallocArray( &pArrayTmplt, &desc,
                               limg.cols, limg.rows) );
CUDART_CHECK( cudaMallocArray( &pArrayImage, &desc,
                               rimg.cols, rimg.rows) );

// Cópia das imagens para memória
CUDART_CHECK( cudaMemcpyToArray( pArrayTmplt, 0, 0,
                                 limg.hData, limg.cols*limg.rows,
                                 cudaMemcpyHostToDevice) );

CUDART_CHECK( cudaMemcpyToArray( pArrayImage, 0, 0,
                                 rimg.hData, rimg.cols*rimg.rows,
                                 cudaMemcpyHostToDevice) );

// Disponibilizando acesso para as texturas
CUDART_CHECK( cudaBindTextureToArray(texTmplt, pArrayTmplt) );
CUDART_CHECK( cudaBindTextureToArray(texImage, pArrayImage) );
```

3.3.2 Nova Estratégia de Crescimento de Regiões

O crescimento de regiões coordena as buscas de homólogos realizando iterações de adensamento sobre os pares de imagens disponíveis no projeto fotogramétrico partindo de sementes inicializadas para preencher as regiões onde haja sobreposição

estereoscópica. Diferente da execução sequencial, onde as buscas de homólogos durante o crescimento de regiões são executadas uma por vez, a execução em paralelo cria a possibilidade de processar diversas buscas simultaneamente.

A busca de homólogos foi seccionada em dois subprocessos. Assim é executado um *kernel* onde cada *thread* é responsável pelo cálculo de um índice de correlação e, posterior a todas as correlações, executa-se outro *kernel* para determinar os máximos e os devidos ajustamentos dos pontos de cada busca. O *kernel* de ajustamento implementado a parte permite ainda a observação da variação do ganho quando a busca é total ou parcialmente executada em GPU, além de permitir averiguar quanto do ganho se deve a aceleração das correlações e quanto se deve a aceleração dos ajustamentos.

No *kernel* de correlações *threads* de um mesmo bloco executam correlações de uma única busca, permitindo o melhor aproveitamento da localidade de memória acessada. Contudo o *grid* utilizado é tridimensional para permitir a realização do empilhamento de diferentes buscas, ou seja, há duas dimensões definidas de modo a comportar uma única busca de homólogos num bloco e uma terceira dimensão é usada para o empilhamento de buscas fazendo ocorrerem tantos blocos quantas forem as buscas a realizar. Isto permite ao algoritmo de crescimento de regiões comandar buscas simultâneas aumentando a ocupação da GPU e reduzindo o número de chamadas ao *kernel* de busca de homólogos. Assim o algoritmo de crescimento de regiões submete para execução em paralelo um *grid* com todas as buscas de homólogos que podem ser processadas a cada iteração.

```
__global__ void make_correlations_kernel(float *P,
                                         size_t pitch,
                                         float N,
                                         controlParameters cp,
                                         homologousAdjustment *seeds)
{

    unsigned char I, T;
    unsigned int SumI = 0, SumISq = 0;
    unsigned int SumT = 0, SumTSq = 0;
    unsigned int SumIT = 0;

    // Endereço do thread no grid
    size_t row = blockIdx.y*blockDim.y + threadIdx.y;
    size_t col = blockIdx.x*blockDim.x + threadIdx.x;
    size_t sid = blockIdx.z; // seed id

    // Ajustando ponteiro para linha e semente correta
    P = (float *) ((char *) P +row*pitch +cp.roiDim *pitch*sid);
```

```
    // Bloqueando threads que superarem os limites de busca
    if ( col >= cp.roiDim || row >= cp.roiDim)
        return;

    // Somatórios para o cálculo de um índice de correlação
    for ( int y = 0; y < cp.tmpltDim; y++ )
    {
        for ( int x = 0; x < cp.tmpltDim; x++ )
        {
            T = tex2D( texTmplt,
                    (float) seeds[sid].tmplt_x+cp.tmpltOffset+x,
                    (float) seeds[sid].tmplt_y+cp.tmpltOffset+y);
            I = tex2D( texImage,
                    (float) seeds[sid].image_x+cp.roiOffset+col+x,
                    (float) seeds[sid].image_y+cp.roiOffset+row+y);
            SumT += T;
            SumI += I;
            SumTSq += T*T;
            SumISq += I*I;
            SumIT += I*T;
        }
    }

    // Cálculo da correlação
    float DenomPart = f(N*SumTSq) - f(SumT*SumT);
    P[col] = correlation_o2( f(SumI), f(SumISq),
                             f(SumIT), f(SumT),
                             N, DenomPart );
}
```

Após todas os índices de correlação serem calculados é necessário determinar os máximos de correlação para cada busca e realizar o ajustamento das coordenadas indicadas por cada semente. Esta tarefa é realizada pelo *kernel* de ajustamento.

A justifica de uma avaliação do ganho em separado desta etapa considera a diferença radical no número de acessos a memória e no total de operações associativas entre o *kernel* de ajustamento e o *kernel* de correlações. Por este motivo, foram implementadas as versões paralela e sequencial para este subprocesso.

A versão paralela do ajustamento atribui a cada *thread* a determinação de máximos e ajustamentos de uma semente. Sua codificação é mostrada a seguir:

```
__global__ void make_adjustment_kernel(float *P, size_t pitch,
                                       controlParameters cp,
                                       homologousAdjustment *seeds){
```

```
    float max = -1;
    int x = 0, y = 0;
    size_t sid = blockIdx.x*blockDim.x + threadIdx.x; // seed id

    if ( sid >= cp.numSeeds)
        return;

    // Ajustando ponteiro para semente correta
    P = (float *) ((char *) P +cp.roiDim*pitch*sid);

    // Busca do máximo
    for ( int i = 0; i < cp.roiDim; i++ )
    {
        for ( int j = 0; j < cp.roiDim; j++ )
        {
            float value = P[i*pitch+j];
            if (value > max)
            {
                max = value;
                x = j;
                y = i;
            }
        }
    }
    seeds[sid].corrMax = max;

    // Ajustamento
    if (max > cp.threshold)
    {
        seeds[sid].adjusted = true;
        seeds[sid].image_x += cp.roiOffset + x;
        seeds[sid].image_y += cp.roiOffset + y;
    }
}
```

A quantidade de sementes usadas no crescimento de regiões aumenta a medida que as sementes de uma iteração anterior sejam bem sucedidas na busca de homólogos. Isto ocorre porque para cada busca bem sucedida é possível lançar até quatro novas sementes a uma distância parametrizável. Reproduzir esta característica impõem uma restrição sequencial entre as iterações do crescimento, contudo todas as sementes bem sucedidas definem uma fronteira de crescimento que pode ser processada em paralelo.

A figura 3.5 ilustra, da esquerda para a direita, como a o número de sementes na fronteira de crescimento aumenta rapidamente a cada iteração de processamento.

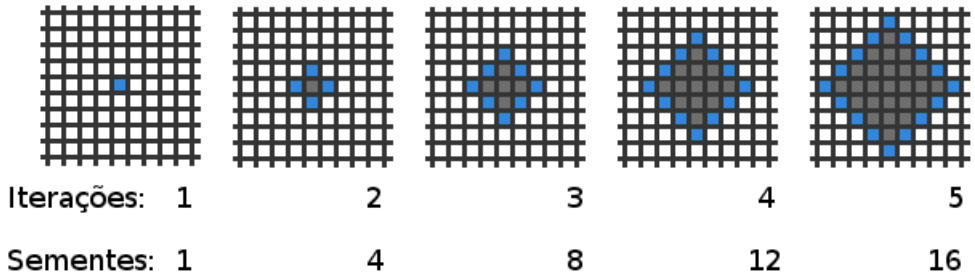

Figura 3.5: Ilustração da fronteira do crescimento de regiões.

Na condição ideal, onde o limiar de correlação é superado em todas as buscas de homólogos para as sementes lançadas, a quantidade de novas sementes na fronteira do crescimento de regiões é uma função do número de iterações, ou seja, partindo de uma semente inicial o crescimento deve atender a seguinte formulação:

$$f(t) = 4t - 4; t > 1 \tag{3.3}$$

É adequado ao processo que as sementes de inicialização sejam distribuídas de forma equidistante na área de sobreposição das imagens, pois as fronteiras de crescimento atingem proporções consideráveis mais rapidamente nesta condição. Contudo, sementes muito próximas dos limites do modelo podem não progredir devido ao valor parametrizável de afastamento das bordas.

De acordo com a distância parametrizável na qual novas sementes podem ser lançadas é definido um mapa de regiões para que possa existir um controle de quais regiões já foram visitadas. O mapa de regiões é demonstrado na figura 3.6. Quem exerce o controle do crescimento é a CPU que memoriza as regiões já preenchidas pelo processamento e coordena o lançamento de novas sementes na direção do crescimento. Nas regiões preenchidas há o cuidado de memorizar que pontos das imagens foram acionados para computar a região.

Uma característica do programa preexistente é a limitação do crescimento imposto quando na execução sequencial uma semente se expande sobre regiões processáveis por outras sementes. Isto marca o mapa de regiões visitadas e impede o crescimento da semente subsequente inviabilizando o aprimoramento do modelo caso as contribuições daquela semente possam ser superiores às atribuídas pelas sementes anteriores.

Para aperfeiçoar este comportamento o programa implementado neste trabalho marca a visita para buscas nas dimensões das imagens e numa grade que corresponde às regiões do modelo segundo o passo de crescimento parametrizado, ou seja, ele grava quais pontos foram usados das imagens para buscar homólogos e quais qua-

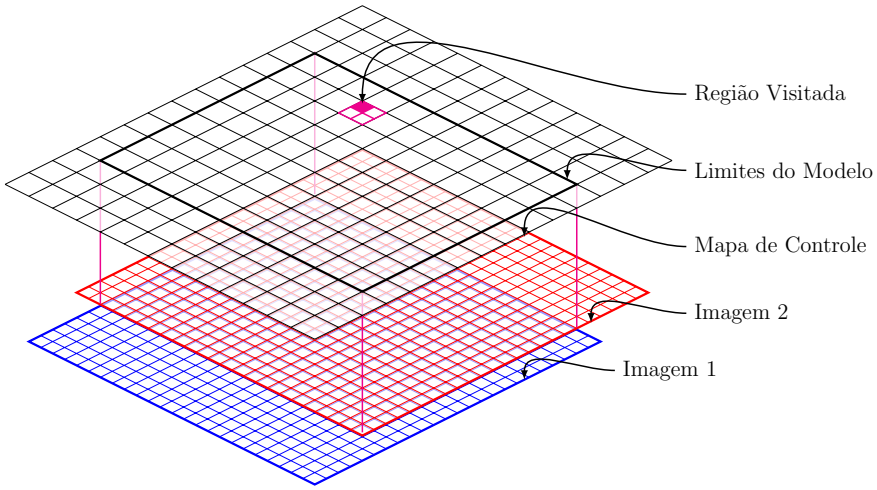

Figura 3.6: Ilustração do mapa de controle de regiões visitadas.

drantes do modelo processável foram cobertos. Assim uma fronteira de crescimento pode lançar buscas dentro de regiões já processadas pela expansão de outra semente, desde que os pontos a serem visitados nas imagens não coincidam plenamente com aqueles visitados por outras sementes.

Quando as fronteiras de crescimento de sementes distintas se encontram, é possível que elas se tornem uma única fronteira de crescimento ou que uma das regiões processadas seja refinada. A união de fronteiras se dá por consenso se as próximas sementes de uma fronteira de crescimento recaírem sobre pontos já visitados por outra. Contudo, quando os pontos estimados não coincidem uma fronteira irá lançar sementes na região já processada para verificar se os índices de correlação obtidos superam àqueles definidos pela fronteira anterior criando assim a possibilidade de refinamentos.

A figura 3.7 exemplifica três caso possíveis de colisão de fronteiras (a), (b) e (c); onde da esquerda para a direita é ilustrado o crescimento partindo de uma iteração n na qual um semente é lançada próximo de regiões já processadas. Uma vez que novas sementes coincidam sobre os mesmos pontos de uma região já processada não ocorrem tentativas de revisão para aperfeiçoar o resultado, ilustrado pela mesma figura na iteração $n + 2$ no caso (a). Se uma fronteira em expansão lança sementes em pontos diferentes daqueles usados ao processar resultados para uma região, então um novo índice de correlação é calculado, como é possível verificar na iteração $n + 2$ nos casos (b) e (c). Quando o novo índice é superior ele substitui o anterior e a fronteira de crescimento avança sobre a área já processada, que é observado nas iterações $n + 3$ e $n + 4$ no caso (c). Caso o índice anterior seja superior o mesmo é preservado e a fronteira de crescimento deixa de progredir através da região, visto nas iterações $n + 3$ e $n + 4$ do caso (b).

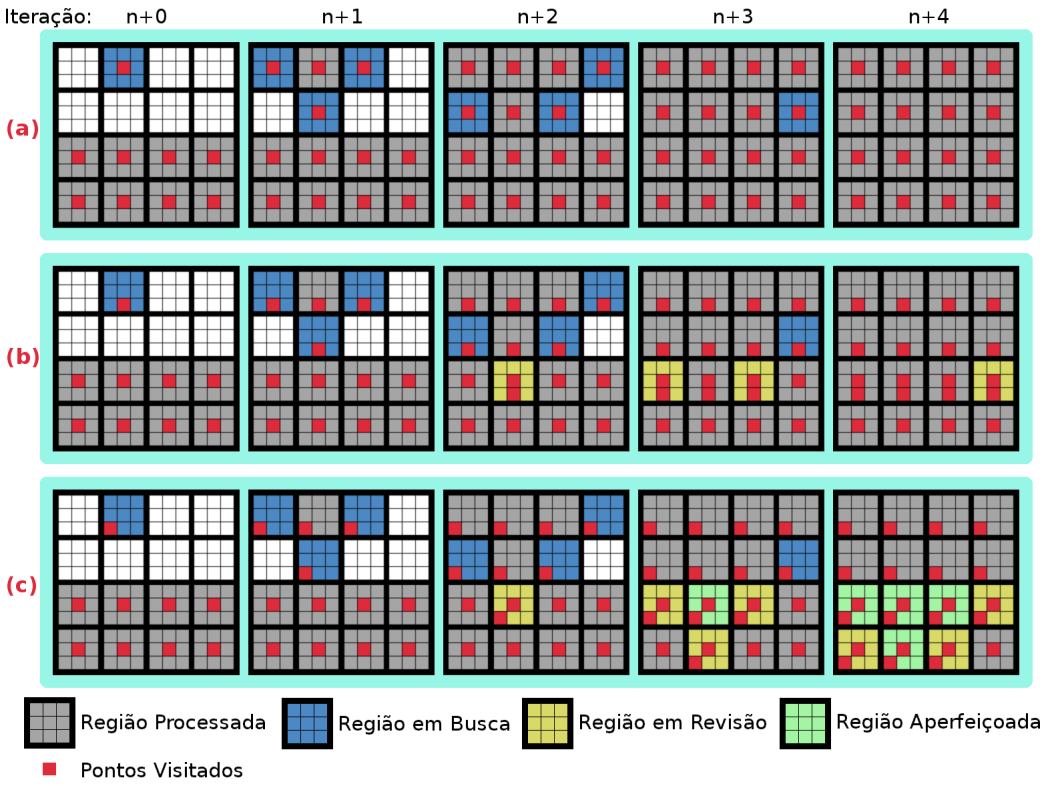

Figura 3.7: Casos de colisão de fronteiras de crescimento: (a) Se sementes coincidem sobre os mesmos pontos; (b) Quando o índice de correlação anterior é superior; (c) Quando o novo índice de correlação é superior.

Capítulo 4

Resultado e discussões

Este capítulo sintetiza os testes realizados na execução deste trabalho e discute os resultados de modo a esclarecer diferenças de desempenho e qualidade da solução proposta em relação ao programa preexistente obtidos pela implementação da solução proposta.

4.1 Ambiente de Teste e Dados Utilizados

Todos os testes foram realizados num computador com processador Intel i5-2500K de 3,3 GHz, com 8GB de memória RAM e com uma placa gráfica nVidia GeForce GTX 560ti com 1GB de memória e 384 núcleos CUDA. O sistema operacional usado foi o Ubuntu 13.10 de 64 bits com a versão 5.5 do toolkit CUDA.

Nos testes foi utilizado o conjunto de dados de exemplo que é disponibilizado pelo Projeto E-Foto. Este projeto fotogramétrico possui cadastro de três imagens de 300 DPI (*Dots Per Inch*) com aproximadamente 2850x2850 *pixels* e 12 pontos de controle. Também foram anexados ao projeto um conjunto de 8 pontos para a checagem de qualidade. Pela sobreposição das imagens de aproximadamente 60% é possível constituir dois pares estereoscópicos no processamento do modelo digital de elevações. As figuras 4.1 e 4.2 ilustram os pares utilizados.

É possível observar no segundo par estereoscópico (figura 4.2) que há diferenças na radiometria das imagens. Uma possível causa para esta variação radiométrica é o envelhecimento e consequente desgaste do material fotográfico utilizado no projeto fotogramétrico. Tratam-se de imagens de câmeras métricas de filme que foram submetidas a digitalização.

Os resultados da compensação radiométrica por *histogram matching* executada pelo programa pode ser vista na figura 4.3. Mesmo com a correção é possível notar diferenças dos níveis de cinza entre as imagens. Contudo esta variação de brilho e contraste perceptível pode ser tolerada pelo algoritmo de correlação utilizado.

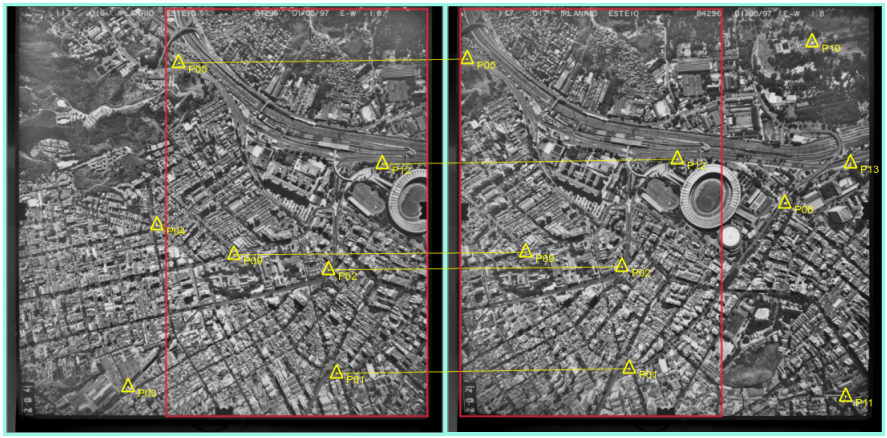

Figura 4.1: Par de imagens estereoscópicas (1/2) e distribuição dos pontos usados como sementes nos testes. As áreas marcadas em vermelho ilustram a sobreposição das imagens.

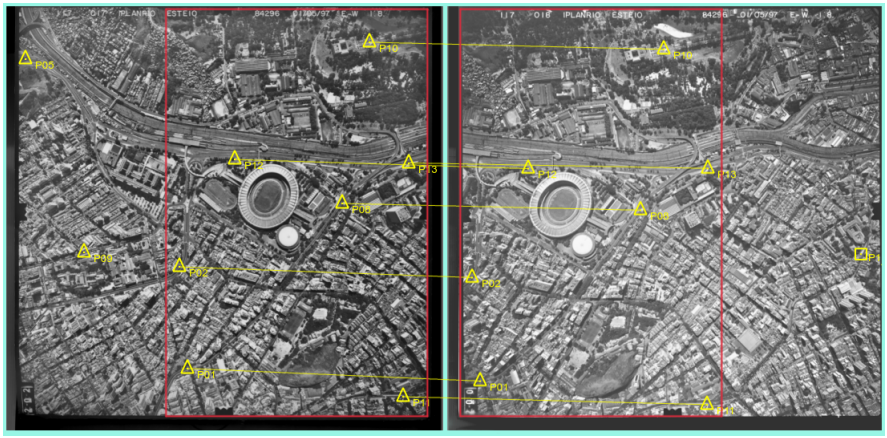

Figura 4.2: Par de imagens estereoscópicas (2/2) e distribuição dos pontos usados como sementes nos testes. As áreas marcadas em vermelho ilustram a sobreposição das imagens.

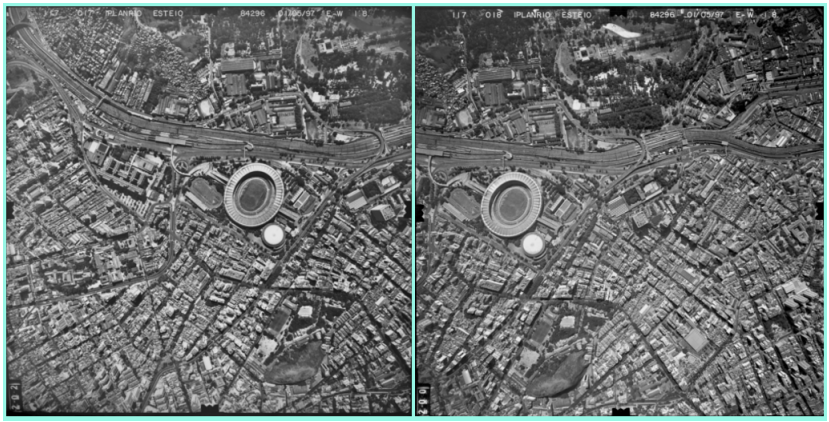

Figura 4.3: Par de imagens estereoscópicas após atuação da correção radiométrica de aproximação do histograma das imagens.

4.2 Tamanho da Janela de Busca

De modo a identificar a janela de busca que maximiza os resultados o algoritmo foi executado para diferentes dimensões. Em todos os casos foram registrados o tempo de execução, número de pontos tridimensionais obtidos e qualidade dos resultados. O resultado pode ser observado na figura 4.4 onde as barras informam o total de pontos no modelo extraído e a linha ascendente demonstra a variação do tempo de execução em função do aumento das dimensões de busca.

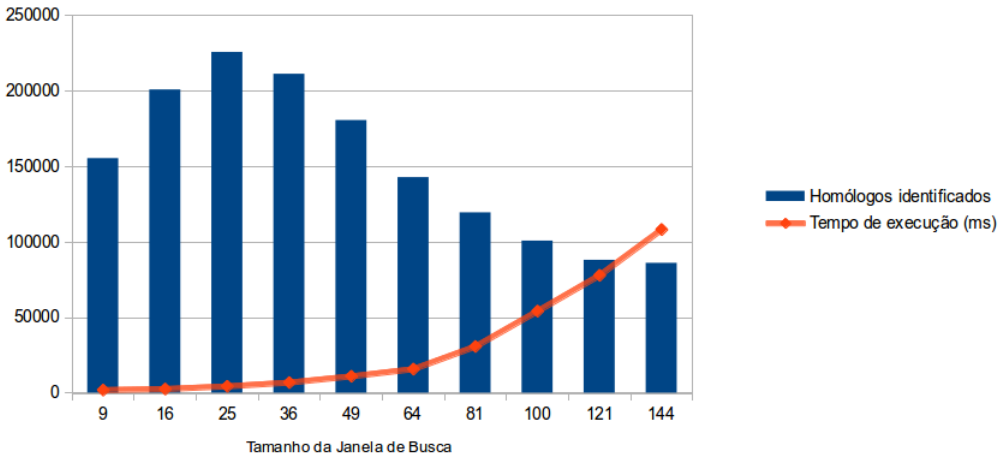

Figura 4.4: Variação de tempos de execução e número de pontos obtidos em função do tamanho das janelas de busca.

Observa-se que embora o tempo de execução possa aumentar demasiadamente em grandes dimensões de busca a qualidade do modelo extraído é deteriorada quando a janela de busca supera um valor ideal. Isto ocorre devido a possibilidade da janela de busca muito grande propiciar a ocorrência de falsos positivos resultando em ajustamentos que degradam as chances de sucesso nas iterações subsequentes do crescimento de regiões. Por esta verificação justifica-se o uso da janela de busca de 25x25 correlações nos testes subsequentes.

4.3 Redução de Chamadas de Função

A modificação no código do programa preexistente foi conduzida no início deste trabalho para determinar o potencial ganho da simplificação da estrutura de dados utilizada para armazenar as imagens durante o processamento. A tabela 4.1 demonstra tempos de execução para o programa preexistente inalterado e com modificações. Nos testes foram executadas extrações para o primeiro par de imagens considerando diferentes escalas de redução das imagens. Observa-se que o número

de homólogos identificados e tempo de execução aumentam conforme aumentam as dimensões das imagens, contudo as quantidades obtidas para cada par de imagens permanece inalterado entre as duas versões e o ganho em aceleração se mantém constante.

Tabela 4.1: Comparação dos tempos de execução para extrair o modelo de um par de imagens entre o programa preexistente inalterado e com modificações.

Escala de redução	Dimensões das imagens	Tempo de execução do prog.		Número de homólogos	Aceleração
		inalterado	c/ modificações		
1/1	2850x2850	2.784,79 s	1.567,01 s	194.449	1,78x
1/2	1425x1425	286,52 s	161,54 s	30.765	1,77x
1/3	950x950	72,92 s	40,94 s	9.717	1,78x
1/4	712x712	20,13 s	11,30 s	2.889	1,78x
1/5	570x570	8,71 s	4,88 s	1.302	1,78x
1/6	475x475	4,91 s	2,76 s	643	1,78x

A tabela 4.2 demonstra os percentuais de tempo dispendido em cada uma das funções durante a execução do programa preexistente com modificações. Trata-se de um resumo do relatório, *flat profile*, criado pelo Gprof utilizando imagens em escala reduzida pelo fator de $\frac{1}{4}$ devido as limitações das variáveis de contagem desta ferramenta.

Tabela 4.2: Perfil do tempo de execução do programa preexistente com modificações de acesso direto à memória.

Tempo (%)	(%) Acum.	Tempo Próprio	Número de Chamadas	Função
40,32	40,32	19,88	112.795.965	Média
38,58	78,90	19,02	56.422.215	Desvio padrão
18,34	97,24	9,04	28.186.875	Covariância
0,91	98,15	0,45	28.186.875	Correlação
0,63	98,78	0,31	45.099	Busca de homólogos
0,16	98,94	0,08	4	Carregamento de imagem
0,16	99,10	0,08	975.364	Alocação de memória
0,10	99,20	0,05	16.284.082	Método de acesso
0,10	99,30	0,05	769.994	Cópia de memória
0,07	99,37	0,04	11.024.241	Escrita em memória
0,02	99,39	0,01	10.681.311	Contagem de colunas
0,02	99,41	0,01	48.465	Validação de *template*
0,02	99,43	0,01	25.664	Triangulação
0,02	99,45	0,01	12	Crescimento de regiões

A classe de cálculos matriciais não foi removida do programa, pois seu uso ainda é fundamental, contudo o uso do método de acesso desta classe foi contornado pelo acesso direto a memória na leitura de dados para o processamento da extração. Isto reduziu o uso do método de acesso em três ordens de grandeza, ou seja, ele passou de aproximadamente 11 bilhões (ver tabela 3.1) para pouco mais de 16 milhões de chamadas, proporcionando assim o aumento de desempenho. Observa-se ainda que assim como sugeriam as expectativas iniciais deste trabalho, se adotadas mudanças

para o uso da estrutura de dados conforme demonstram tais resultados, as funções com maior participação no tempo de execução são aquelas diretamente associadas ao cálculo de cada índice de correlação.

É importante também identificar quais funções demandam maior tempo de resposta por chamada considerando as ramificações de processamento encadeadas por sua execução. Neste contexto a função de crescimento de regiões se destaca como visto na figura 4.5, que apresenta o grafo de chamadas com ramificações encadeadas a este método. Somada a representatividade dos tempos de execução de cada função invocada pelo método de crescimento de regiões acumula-se 98,8% de todo o tempo de execução. Contudo, tal método executa poucas vezes, sendo um total de 12 chamadas que constitui o número de sementes utilizadas para inicialização do modelo, isto é, 5 sementes no primeiro par de imagens e 7 sementes no segundo par.

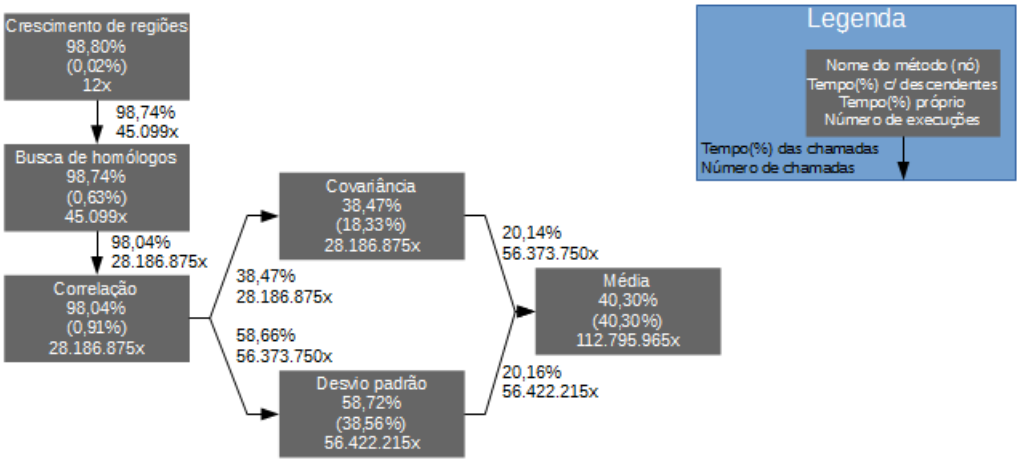

Figura 4.5: Visão da ramificação que dispende maior tempo de execução após modificação do programa preexistente. Adaptada do relatório, *call graph*, gerado com o *Gprof*.

4.4 Análise de Cobertura

O aperfeiçoamento da estratégia de crescimento de regiões adotado pela solução proposta neste trabalho resultou no preenchimento de regiões que não eram alcançadas pelo programa preexistente. Conforme mostra a figura 4.6 algumas edificações que não eram cobertas passaram a ser preenchidas. Ao todo a cobertura da área total do modelo passou de 53.5% na solução anterior para 68.5% na solução proposta neste trabalho.

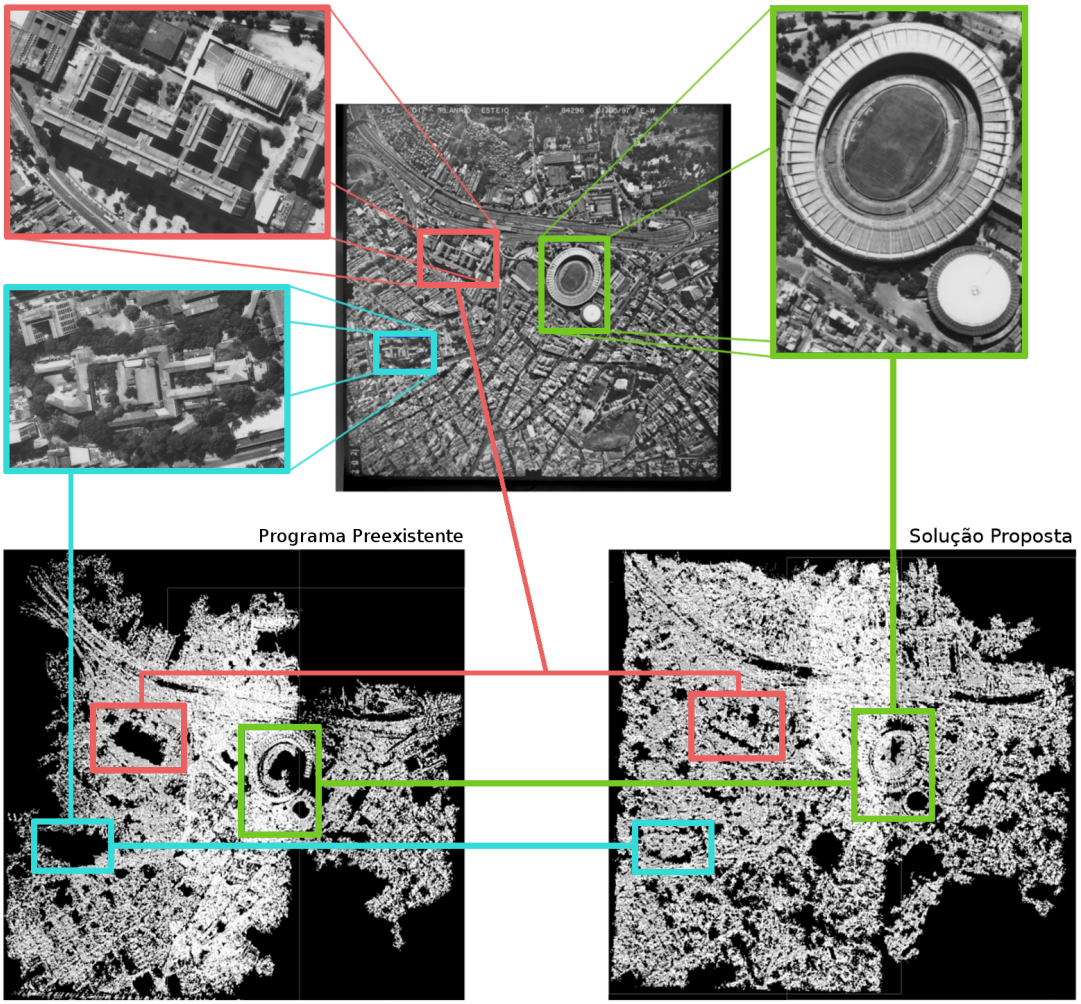

Figura 4.6: Comparativo da cobertura em edificações muito elevadas ou cercadas por regiões de difícil correlação.

De modo a verificar que no programa preexistente uma semente pode impedir o crescimento de demais sementes foram computados os totais de buscas de homólogos executadas por cada semente conforme mostra a tabela 4.3. Observa-se no primeiro par de imagens que a semente utilizada na primeira chamada do crescimento de regiões impede o uso das demais. Isso não ocorre no segundo par devido a limitações impostas ao modelo como, por exemplo, a parametrização de bordas, o relevo e as diferenças radiométrica que colaboram para a interrupção do crescimento de regiões das sementes iniciais antes do preenchimento das regiões onde estão alocadas outras sementes.

Os testes de cobertura incluíram o registro de evolução do preenchimento para os métodos de crescimento usados. A variação na forma como a fronteira de crescimento é preenchida está diretamente ligada a estrutura de dados usada, mas é também justificável pela execução em paralelo do processamento de buscas.

Tabela 4.3: Comparação do número de buscas de homólogos disparadas por cada semente para o programa preexistente e a solução proposta. É evidenciado o bloqueio do crescimento que ocorre no programa preexistente.

Par	Semente	Total de Buscas Derivadas	
		Programa preexistente	Solução proposta
1	1	361.823	41.633
1	2	0	77.802
1	3	0	0
1	4	0	112.003
1	5	0	121.172
2	1	0	0
2	2	0	0
2	3	102.163	90.881
2	4	0	0
2	5	0	0
2	6	77.950	83.898
2	7	0	24.593

O método de crescimento de regiões implementado no programa preexistente é baseado no algoritmo de *floodfill* de 4 direções utilizando pilhas. Seu resultado é um crescimento que progride prioritariamente em uma direção até que uma correlação falhe. Isto traça linhas adjacentes conforme uma linha encontra seu limite e outras linhas começam a ser traçadas nas proximidades de onde a anterior parou. Devido ao ajustamento que ocorre na busca de cada ponto homólogo o resultado observado quando considerado o traçado do crescimento não necessariamente forma linhas paralelas. A figura 4.7 demonstra resultados parciais enquanto o crescimento evolui. O número de iterações é elevado, pois no programa preexistente a cada iteração do crescimento de regiões só é executada a busca de homólogos para um par de pontos.

O solução proposta neste trabalho resulta num comportamento de crescimento semelhante ao preenchimento por *floodfill* de 4 direções utilizando uma fila. Isto se deve a capacidade de executar múltiplas buscas de homólogos na versão paralela, onde são enviadas para o processamento na GPU toda as sementes da fronteira de regiões conhecidas. Deste modo os dados para processamento acabam enfileirados entre cada iteração do crescimento de regiões em paralelo. A figura 4.8 ilustra o comportamento da solução proposta onde o crescimento de regiões executa para diversas sementes de forma simultânea.

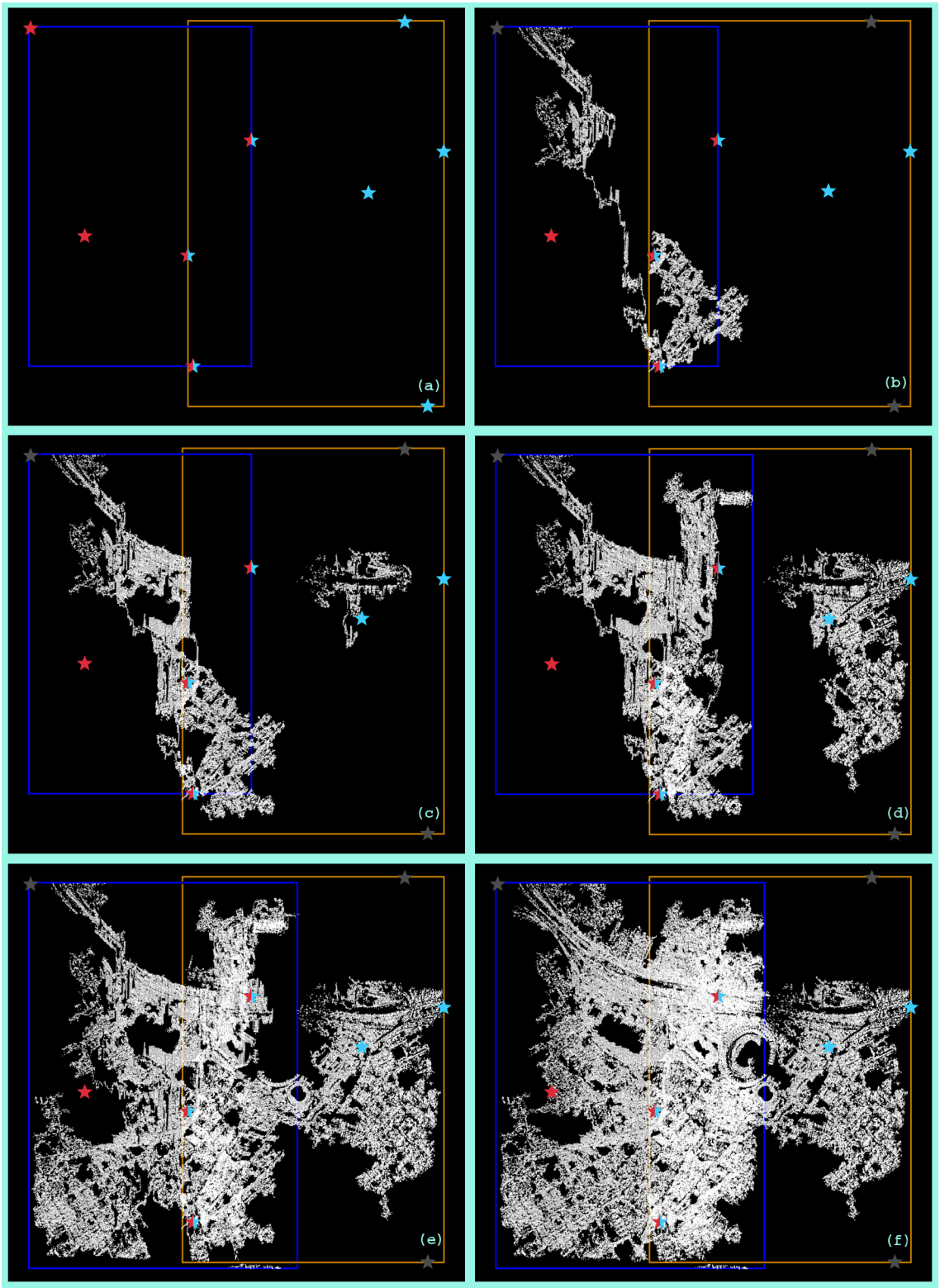

Figura 4.7: Evolução do crescimento de regiões no programa preexistente: (a) Distribuição das sementes; (b) 64k iterações; (c) 128k iterações; (d) 256k iterações; (e) 512k iterações; e (f) Resultado final.

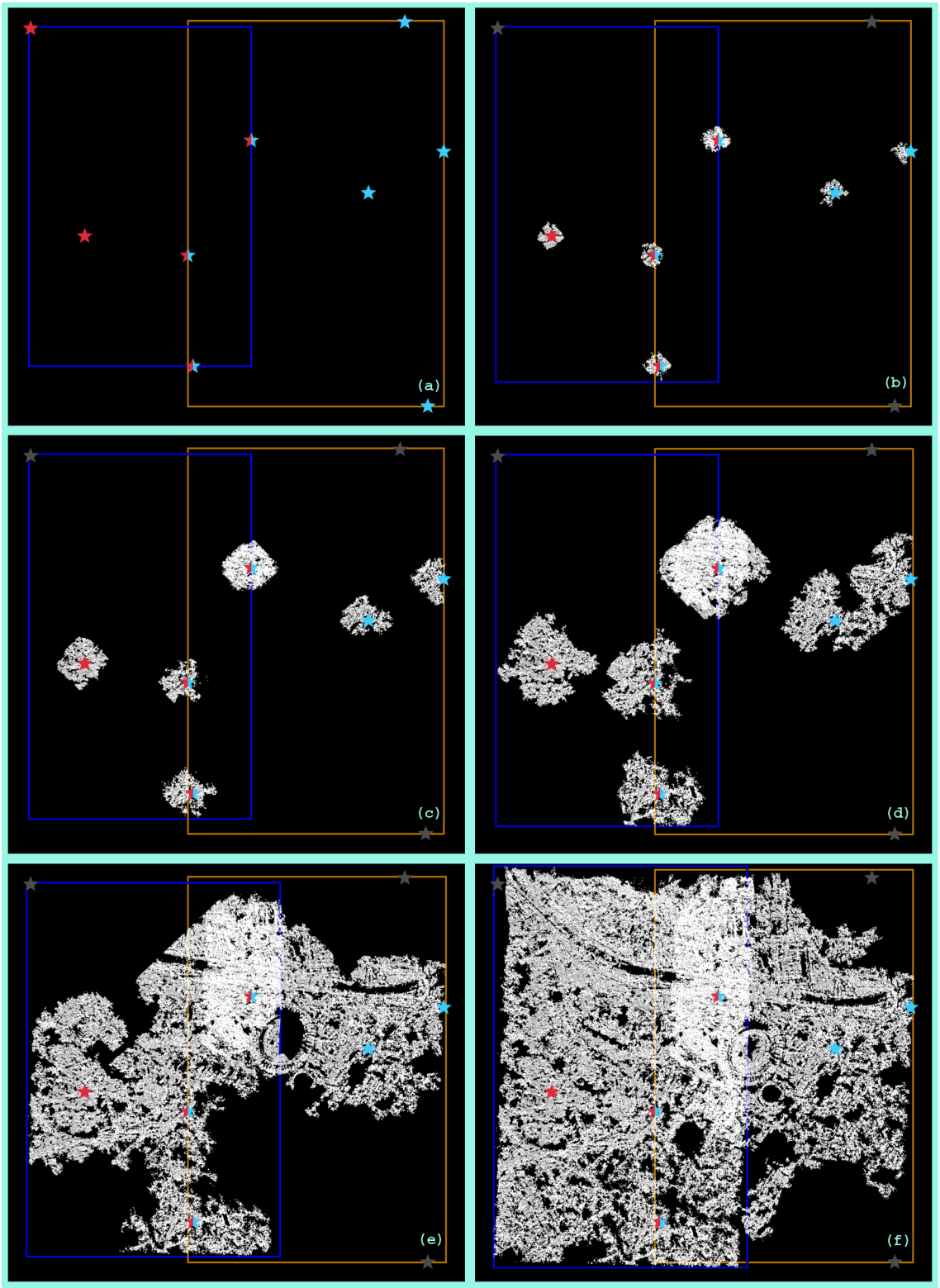

Figura 4.8: Evolução do crescimento de regiões na solução proposta: (a) Distribuição das sementes; (b) 32 iterações; (c) 64 iterações; (d) 128 iterações; (e) 256 iterações; e (f) Resultado final.

4.5 Qualidade dos Resultados

Para a aplicação de mapeamento topográfico a precisão de modelos digitais de elevação pode admitir erros da ordem de metros. Contudo o avanço dos sensores disponíveis pode levar a precisão prática à ordem de centímetros quando trata-se de levantamento aéreo com imagens de resolução igual ou superior a 1200 DPI.

Considerado que a resolução das imagens de exemplo usadas neste trabalho é 4 vezes inferior, os limites da precisão devem manter-se na ordem de metros. Nesta resolução de imagens, se considerada a altura de voo de 1300 metros, é possível determinar a representatividade das dimensões do *pixel* em aproximadamente 0,7 metros.

Embora os recursos disponíveis no teste limitem a qualidade do modelo obtido é possível comparar as precisões entre diferentes técnicas para a extração de modo a demonstrar possíveis variações na qualidade dos resultados. A tabela 4.4 demonstra um comparativo das diferenças altimétricas encontradas quando usados os pontos de checagem de qualidade. A coluna "Preexistente"representa o programa preexistente enquanto a coluna "Proposta"representa a solução proposta neste trabalho, ambos com base no algoritmo de NCC. A título de comparação foi também incluído na análise o algoritmo de LSM (*Least Square Matching*) que representa outra possível solução para correlações de homólogos.

Tabela 4.4: Qualidade dos resultados. Comparativo do programa preexistente com o método proposto neste trabalho. Um algoritmo alternativo que implementa correlação por *Least Square Matching* foi inserido a título de referência do quanto é possível melhorar a qualidade.

Coordenadas de Verificação			Erro em Z		
X	Y	Z	Preexistente	Proposta	por LSM
681.100,209	7.464.305,984	12,002	8,584	1,802	——
681.079,091	7.464.791,902	10,103	7,663	6,734	3,961
681.875,866	7.465.066,854	5,980	1,502	1,437	0,186
680.626,292	7.464.876,497	12,075	——	5,191	1,978
682.031,497	7.465.841,798	21,146	13,719	——	——
682.134,395	7.464.127,842	8,656	5,946	——	——
681.361,203	7.465.299,251	7,796	0,421	0,376	0,526
682.208,231	7.465.250,939	5,913	——	1,756	2,178
		Erro médio	6,305	2,882	1,765
		Desvio Padrão	4,895	2,488	1,505

Esta qualidade foi computada a partir de diferenças altimétricas entre pontos de qualidade e os modelos. Isto é possível pelo uso de interpolação bilinear para computar qual a tendência da superfície modelada ao representar as mesmas coordenadas planimétricas. Contudo a indisponibilidade de vizinhos próximos identificáveis nos modelos fazem com que seja impossível determinar a qualidade para todos os pontos ocasionando a ausência de alguns valores na tabela.

Observa-se que os valores de erro médio e desvio padrão melhoraram consideravelmente em relação ao algoritmo de NCC usado no programa preexistente. Este fato pode estar relacionado ao melhor aproveitamento das sementes ao permitir o aperfeiçoamento de regiões durante o crescimento. O resultado do algoritmo de LSM apresenta melhor qualidade final devido a sua capacidade de compensar, com transformações afim, as deformações de rotações e escalas não suportadas pelo NCC.

4.6 Tempos de Execução

Os tempos de execução obtidos durante os testes para determinar a aceleração estão descritos na tabela 4.5. Partindo do programa preexistente inalterado foram evidenciados os ganhos em aceleração incremental e acumulada ao comparar o programa preexistente com modificações e a solução proposta deste trabalho. A solução proposta é apresentada para os dois modos de execução possíveis (serial e paralelo). Os tempos de execução registrados se referem à extração de modelos com um e dois pares de imagens, utilizando a dimensão constante de 25x25 na parametrização da janelas de busca.

Tabela 4.5: Comparação dos tempos de execução e determinação da aceleração obtida entre o programa preexistente e a solução proposta. Teste executado com janelas de busca de 25x25, com 1 e 2 pares de imagens e sem redução de escala nas imagens.

	Tempo de execução (s)		Aceleração	
Versão	c/ 1 par	c/ 2 pares	Incremental	Acumulada
Programa preexistente inalterado	4648,04 s	2784,79 s	——	——
Programa preexistente c/ modificações	2505,84 s	1567,01 s	1,8x	1,8x
Solução proposta no modo serial	68,62 s	42,21 s	36,9x	66,4x
Solução proposta no modo paralelo	7,35 s	4,50 s	9,4x	624,3x

Medições do tempo de execução da solução proposta foram efetuadas também para demonstrar a participação de cada um dos subprocessos no tempo total de execução conforme mostra a tabela 4.6.

Tabela 4.6: Comparação dos tempos de execução e determinação da aceleração obtida entre os modos de execução da solução proposta. Teste executado com janelas de busca de 25x25, com 2 pares de imagens e sem redução de escala nas imagens.

	Tempo de Execução (ms)		
Etapa	em Serial	em Paralelo	Aceleração
Preparo de imagens	335,8	——	——
Troca de Dados CPU-GPU e GPU-CPU	——	2.496,5	——
Calculo de correlações	59.538,0	176,3	337,7x
Seleção de Máximos e Ajustamentos	1.169,4	769,0	1,5x
Definição da Fronteira de Busca	56,9	——	——
Triangulação de coordenadas 3D	2.748,2	——	——
Gravação dos resultados	681,4	——	——

Capítulo 5

Conclusão e Trabalhos Futuros

Segundo demonstrado no capítulo anterior, as melhorias implementadas em relação ao programa preexistente constituem-se por ganhos em qualidade e aceleração da produção de resultados. O aumento da qualidade é também relacionado à melhoria da capacidade de cobertura na produção dos modelos digitais de elevação e o melhor aproveitamento das semente de inicialização. Enquanto que os fatores responsáveis pela aceleração são a modificação no uso de estruturas de dados, a otimização da formulação de correlação e a paralelização em GPU.

A aceleração obtida pela paralelização em GPU demonstra o potencial de aumento de desempenho quando utilizados estes dispositivos. Contudo, o ganho relativo de desempenho já na execução serial do algoritmo é consideravelmente maior. Isto evidência um grande impacto positivo dos aperfeiçoamentos de rotinas e simplificações de estruturas executados ao longo do desenvolvimento deste trabalho.

Uma consideração a ser feita deve-se ao tempo necessário para executar transferência de dados entre a GPU e CPU que é adicionado ao tempo de execução total na vertente de processamento paralelo e a necessidade de cooperação com a CPU na definição de novas sementes. Tais requisitos contrabalanceiam com o ganho por busca individual de homólogos.

Um cenário que possivelmente poderia equilibrar a diferença de ganhos é o trabalho com imagens de resoluções maiores. Neste caso, o número de buscas processadas aumenta significativamente e o tamanho da melhor janela de busca aplicável também tende a aumentar. Por consequência, o ganho geral tende a aumentar.

Trabalhos futuros em caráter multidisciplinar podem partir dos resultados desenvolvidos nesta dissertação. Como investigações a respeito da execução com imagens de maiores resoluções, o aperfeiçoamento de extratores de modelos digitais baseados em correlação pelo algoritmo de *Least Square Matching* ou a investigação da aplicação a imagens de curta distância ou imagens de níveis orbitais. Por fim, pode ocorrer a integração a demais soluções de reconstrução de cenas para viabilizar um processo mais amplo de modelagem.

Referências Bibliográficas

[1] GISRESOURCE. "Difference between DEM/DTM and DSM". July 2014. Disponível em: <http://www.gisresources.com/confused-dem-dtm-dsm/>.

[2] MILLER, C. *The Spatial Model Concept of Photogrammetry*. Publication (Massachusetts Institute of Technology. Photogrammetry Laboratory). Massachusetts, M.I.T. Photogrammetry Laboratory, 1957. Disponível em: <http://books.google.com.br/books?id=JGs7HQAACAAJ>.

[3] MILLER, C., LAFLAMME, R. *The Digital Terrain Model -: Theory & Application*. Publication (Massachusetts Institute of Technology. Photogrammetry Laboratory). Massachusetts, M.I.T. Photogrammetry Laboratory, 1958. Disponível em: <http://books.google.com.br/books?id=qF30GwAACAAJ>.

[4] CCEA. "Computing in Civil Engineering Award". July 2014. Disponível em: <http://www.asce.org/leadership-and-management/awards/computing-in-civil-engineering-award/>.

[5] OCHI, S., LIZUKA, T., HAMASAKI, E. *Charge-Coupled Device Technology*. Japanese technology reviews (Gordon and Breach): Electronics. Amsterdam, Taylor & Francis, 1997. Disponível em: <http://books.google.com.br/books?id=fVJ5txTE0mIC>.

[6] COELHO, L., BRITO, J. N. *Fotogrametria Digital*. 2 ed. Rio de Janeiro, EdUERJ, 2007. ISBN: 9788575111147. Disponível em: <http://www.efoto.eng.uerj.br/images/stories/Livro/fotogrametria_digital_revisado.pdf>.

[7] HARTLEY, R. I., ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. Second ed. Cambridge, Cambridge University Press, 2004. ISBN: 0521540518. Disponível em: <http://www.robots.ox.ac.uk/~vgg/hzbook/index.html>.

[8] USGS. “USGS SDTS format Digital Elevation Model data (DEM)”. July 2014. Disponível em: <http://data.geocomm.com/dem/>.

[9] DTED. “Digital Terrain Elevaion Data [DTED]”. July 2014. Disponível em: <http://fas.org/irp/program/core/dted.htm>.

[10] DIMAP. “Digital Image MAP v1.1”. July 2014. Disponível em: <http://www.spotimage.com/dimap/spec/dimap.htm>.

[11] CIPOLLA, R., BATTIATO, S., FARINELLA, G. *Computer Vision: Detection, Recognition and Reconstruction*. Studies in Computational Intelligence. Catamia, Springer, 2010. ISBN: 9783642128479. Disponível em: <http://books.google.com.br/books?id=8yVnm_PYSv0C>.

[12] SHEPPARD, A. *Programming GPUs*. Oreilly and Associate Series. Califórnia, O'Reilly Media, Incorporated, 2013. ISBN: 9781449302351. Disponível em: <http://books.google.com.br/books?id=AJWUZwEACAAJ>.

[13] E-FOTO. “O que é o e-foto?” July 2014. Disponível em: <http://www.efoto.eng.uerj.br/br/o-que-e-o-e-foto>.

[14] NVIDIA. “GeForce GTX 560 ti”. July 2014. Disponível em: <http://www.nvidia.com.br/object/product-geforce-gtx-560ti-br.html>.

[15] SOBELL, M. *A Practical Guide to Ubuntu Linux*. Michigan, Pearson Education, 2010. ISBN: 9780132483933. Disponível em: <http://books.google.com.br/books?id=k9bHSd97HzMC>.

[16] GOUGH, B., STALLMAN, R. *An Introduction to GCC: For the GNU Compilers Gcc and G++*. A Network theory manual. Bristol, Network Theory, 2004. ISBN: 9780954161798. Disponível em: <http://www.network-theory.co.uk/docs/gccintro/gccintro_81.html>.

[17] WADLEIGH, K., CRAWFORD, I. *Software Optimization for High-performance Computing*. HP Professional Series. Neew Jersey, Prentice Hall PTR, 2000. ISBN: 9780130170088. Disponível em: <http://books.google.com.br/books?id=IRN0IEXJzKEC>.

[18] ELLIS, M., STROUSTRUP, B. *C++: manual de referencia comentado*. São Paulo, Campus. Disponível em: <http://books.google.com.br/books?id=RYBJkgEACAAJ>.

[19] WILT, N. *The CUDA Handbook: A Comprehensive Guide to GPU Programming*. Indiana, Pearson Education, 2013. ISBN:

9780133261509. Disponível em: <http://books.google.com.br/books?id=ynydqKP225EC>.

[20] BLANCHETTE, J., SUMMERFIELD, M. *C++ GUI Programming with Qt 4*. Massachusetts, Prentice Hall, 2006. ISBN: 9780131872493. Disponível em: <http://books.google.com.br/books?id=tSCR_4LH2KsC>.

[21] TOUTIN, T. "Elevation modelling from satellite visible and infrared (VIR) data", *International Journal of Remote Sensing*, v. 22, n. 6, pp. 1097–1125, 2001. doi: 10.1080/01431160117862. Disponível em: <http://dx.doi.org/10.1080/01431160117862>.

[22] RTK. *Using RTK GPS Method in Creation of Digital Terrain Models*, Borovets, January 2006. International conference on cartography and GIS. Disponível em: <http://www.researchgate.net/profile/Ibrahim_Tiryakioglu/publications>.

[23] NOAA. "LIDAR — Light Detection and Ranging — is a remote sensing method used to examine the surface of the Earth." July 2014. Disponível em: <http://oceanservice.noaa.gov/facts/lidar.html>.

[24] YU, W., CHEN, T., FRANCHETTI, F. "High performance stereo vision designed for massively data parallel platforms", *Circuits and Systems for Video Technology, IEEE Transactions on*, v. 20, n. 11, pp. 1509–1519, 2010.

[25] KOWALCZUK, J., PSOTA, E. T., PÉREZ, L. C. "Real-time stereo matching on CUDA using an iterative refinement method for adaptive support-weight correspondences", *Circuits and Systems for Video Technology, IEEE Transactions on*, v. 23, n. 1, pp. 94–104, 2013.

[26] LIU, X., GAO, W., HU, Z.-Y. "Hybrid Parallel Bundle Adjustment for 3D Scene Reconstruction with Massive Points", *Journal of Computer Science and Technology*, v. 27, n. 6, pp. 1269–1280, 2012.

[27] STOCKBURGER, D. *Introductory Statistics: Concepts, Models, and Applications*. Missoure, Cengage Learning, 2007. ISBN: 9781931442466. Disponível em: <http://books.google.com.br/books?id=WSRKPwAACAAJ>.

[28] ARUNAGIRI, S., JALOMA, J. "Parallel GPGPU stereo matching with an energy-efficient cost function based on normalized cross correlation". In: *IS&T/SPIE Electronic Imaging*, pp. 86550X–86550X. International Society for Optics and Photonics, 2013.

[29] SIFT. “SIFT Keypoints detector”. July 2014. Disponível em: <http://www.cs.ubc.ca/~lowe/keypoints/>.

[30] GPGPU. “General-Purpose Computation on Graphics Hardware”. July 2014. Disponível em: <http://gpgpu.org/>.

[31] TITAN Z. “Placa de Vídeo GeForce GTX Titan Z”. July 2014. Disponível em: <http://www.nvidia.com.br/object/geforce-gtx-titan-z-br.html>.