5
Conclusão e Trabalhos Futuros

O texto desta tese foi elaborado na tentativa de formalizar uma introdução à programação em placas gráficas de algoritmos de visão computacional nos níveis mais baixos da hierarquia proposta por Marr para representação do conteúdo visual. Nesta seção apresentamos trabalhos futuros que se beneficiam das contribuições apresentadas na tese e uma revisão conclusiva de tais contribuições. Como linha de pesquisa, a continuação do trabalho desenvolvido aponta também para algoritmos nas demais camadas da hierarquia.

Na Seção 3.3 e no anexo A apresentamos MOCT, uma técnica que propõe procedimentos em GPU para a localização e acompanhamento de trajetórias no espaço da imagem de um conjunto de objetos identificáveis por suas cores. O objetivo do MOCT é encontrar para cada objeto identificado seu centróide, sua massa em número de pixels e sua caixa envolvente ("bounding box") – propriedade que viabiliza um melhor tratamento pelas aplicações aos objetos com áreas não quadrangulares (exemplo: composição de vídeo sobre objetos de formatos diversos).

O algoritmo descrito é composto de uma primeira etapa na qual são efetuadas análises locais em trechos da imagem, uma etapa de aglomeração dos resultados locais em resultados globais e um procedimento para o armazenamento das trajetórias de cada objeto. Mostramos também que a base do algoritmo MOCT pode ser vista como uma adaptação de um operador paralelo de redução múltipla.

Ao formalizar o operador de redução múltipla na Seção 3.1.2, motivados pela reutilização dos dados a serem analisados pelos diferentes operadores, incentivamos seu uso em computações que envolvem a aplicação de um conjunto de operadores sobre uma imagem (um mesmo fluxo), em alternativa a aplicação de um operador por vez. O desempenho do operador de redução múltipla foi avaliado em variações que afetam o número total e os padrões de acessos à textura e a ocupação dos multi-processadores. Embora os resultados de medições de tempo apresentados traduzam a medição de uma placa gráfica específica e seguindo a adaptação do MOCT, as avaliações comprovam o ganho obtido pela utilização de diferentes fatores de redução e o ganho pela adoção

das variações de arranjos propostas, as quais resultam em diferentes padrões de acesso à textura.

Entre as extensões da pesquisa proposta para o operador de redução múltipla, o algoritmo para computação de histograma proposto por Fluck et al. (Flu06) é um forte candidato a ser beneficiado pelas variações de arranjos e avaliações de desempenho obtidas nesta tese. Sua implementação e avaliação seguindo as variações de arranjos e fatores de redução é planejada como trabalho futuro desta tese.

Ainda sobre a divulgação do operador de redução múltipla, esta pesquisa pode ser estendida pelo desenvolvimento de outros exemplos para seu uso. Entre as extensões planejadas a serem desenvolvidas em trabalhos futuros, mantendo o contexto de algoritmos nas camadas de Processamento Baixo Nível e Detecção de Forma, está a implementação de uma transformada de Hough em GPU para o reconhecimento de objetos, tais como retas, circunferências e elipses. A idéia é estabelecer o conjunto de operadores do redutor múltiplo como sendo composto dos diferentes parâmetros do objeto desejado, organizados em um arranjo. Inicialmente, avaliações locais da imagem seriam armazenadas contendo os contadores das diferentes parametrizações. O redutor seria aplicado para formar a matriz acumuladora da transformada, contendo as avaliações globais. Para completar esta proposta de trabalho futuro, tal implementação deve ser comparada com outras soluções para a computação da transformada de Hough em GPU.

Sobre a computação do redutor múltiplo usando linguagens de programação genérica em GPU, fica como outra fonte de investigação, a implementação do gerenciamento dos dados compartilhados entre os diversos operadores aplicados em um subfluxo, pela codificação de pré-carregamentos em memória compartilhada dos dados consultados por cada arranjo do redutor.

Na Seção 3.3, o operador de redução regional múltipla é descrito apresentando uma distribuição do processamento de regiões de interesse em GPU. Em sua formulação, cada região de interesse é repartida em subregiões independentes, as quais são processadas em paralelo para gerar resultados parciais. Em uma segunda etapa do operador, os resultados das subregiões são aglomerados para descrever a análise completa das regiões de interesse. Em comparação ao operador de redução múltipla, o regional evita a avaliação de toda a imagem (ou todo o fluxo de entrada) por cada operador a ser processado, quando é conhecida uma região de interesse descrevendo a área da imagem (ou intervalo do fluxo de entrada) para aplicação de cada operador.

As extensões naturais para o operador regional proposto incluem: a formalização do algoritmo em mais dimensões, viabilizando seu uso no tratamento

de dados volumétricos e sua formalização para regiões de interesse definidas por outras curvas/superfícies, tais como o círculo (2D) e a esfera (3D).

No anexo B, apresentamos uma aplicação do operador regional proposto, descrevendo seu uso durante a computação do Diagrama de Voronoi Centroidal. O operador de redução regional múltipla viabiliza a computação em GPU de um algoritmo clássico para a construção deste tipo de diagrama, conhecido como algoritmo de Lloyd. Ao permitir o cálculo em GPU dos centróides de cada região de Voronoi, o operador regional proposto elimina a necessidade de transferir, entre a CPU e a GPU, as texturas descrevendo os diagramas de Voronoi computados ao longo do algoritmo. Além disso, mostramos que a utilização de regiões de interesse no cálculo dos centróides permite que a eficiência do operador seja mantida, ainda que sejam considerados grandes conjuntos de pontos geradores do diagrama.

No Capítulo 4 e no anexo C propomos um novo modelo de representação das relações de vizinhança entre as folhas de uma quadtree, denominado *QuadN4tree*. A inspiração para este modelo está na observação de que os dados comumente armazenados em uma Quadtree, são representados em seus nós folhas, induzindo à necessidade de métodos de navegação entre elas. Este modelo descreve uma estrutura para a representação das folhas que armazena um quarteto de referências para outras folhas. As referências são especialmente escolhidas para a permitir a navegação entre folhas vizinhas, passando de folha em folha, sem visitar nós internos da árvore, de maneira a cobrir o espaço representado pela quadtree. Entre as propriedades do modelo, mostramos que fornece um mecanismo de consulta ótima para o levantamento das folhas vizinhas de uma folha qualquer. No mesmo capítulo, a Seção 4.5 apresenta a construção da QuadN4tree em GPU.

Interessantes extensões para a QuadN4tree incluem a proposição de algoritmos cobrindo uma maior variedade de operações sobre tal estrutura, tais como: o levantamento de componentes conexos ou regiões maximais, segundo o critério de semelhança utilizado; operações de janelamento, retornando quais folhas fazem intersecção com uma região; operações sobre árvores distintas (união, diferença, interseção).

Em três dimensões, as regras de subdivisão espacial de uma quadtree formam a estrutura conhecida como octree. Outra continuação da pesquisa, a ser concluída em um trabalho futuro é a formalização da extensão do modelo QuadN4 para octrees, na qual seis referências para folhas vizinhas são cuidadosamente escolhidas, de forma a cobrir o volume representado pela octree.

Sobre algoritmos de processamento de imagens e visão computacional

formulados utilizando a técnica de minimização de energia usando corte de grafos, nas seções 4.6 e 4.7 e no anexo D, propomos abordagens em pré-processamento à minimização para acelerar o cálculo de tais aplicações.

Uma primeira contribuição a esse tipo de otimização foi proposta na Seção 4.6.3, a partir da análise da formulação geral da função de energia nesse tipo de otimização. Inicialmente foi observada a independência existente entre as parcelas dos termos da função de energia, justificada pela separação em pesos de diferentes tipos de arestas no grafo. Dois níveis de independência foram destacados: entre termos da função de energia e entre elementos. Com base em tais observações, sugerimos a computação em GPU das diferentes parcelas da função de energia em diferentes passadas, cada passada operando em paralelo sobre os elementos de uma imagem. Tal separação e computação em GPU foi exemplificada para a aplicação de segmentação objeto/fundo por luz ativa.

Em outra contribuição aos algoritmos formulados com esse tipo de otimização, propomos a redução do tamanho do grafo (de seu número de nós e arestas) representando o problema a ser otimizado. Com esta finalidade é proposta a construção de um grafo operando sobre elementos de um particionamento do conjunto de pixels em grupos homogêneos, de acordo com um critério de similaridade coerente à aplicação, representado por uma quadtree de regiões. Nossa motivação para o uso da quadtree é o de tal estrutura fornecer partições de propriedades bem definidas, especialmente as descrições de área, perímetro e vizinhança. A proposta, denominada QuadCut, é apresentada na Seção 4.7 com uma formulação geral para a função de energia, tendo as folhas da quadtree como variáveis da otimização e com a descrição da construção de um grafo sobre as folhas de uma quadtree representando a otimização de uma aplicação genérica.

Como exemplo de aplicação, apresentamos o problema da segmentação objeto/fundo com uso de luz ativa. Os resultados dessa aplicação ilustram a classificação apropriada de detalhes finos da imagem original utilizando o método de QuadCut (Figura 4.7.1). Os detalhes são mantidos, uma vez que, ao mesmo tempo que fornece uma redução ao total de variáveis avaliadas pela otimização, o QuadCut cria folhas tão pequenas quanto 1×1 pixel, se forem necessárias e a elas associa pesos coerentes à sua classificação.

A formulação geral proposta pelo QuadCut pode ser diretamente aplicada em trabalhos futuros a outros algoritmos de segmentação via corte de grafo com abordagens diferentes da adotada (Rot04, Yin04, Wan05). O QuadCut também é extensível às aplicações para classificações entre múltiplos rótulos, tais como, algoritmos de coloração e restauração de imagens propostos respectivamente

em (Yun06) e (Boy98, Raj05). Outra extensão desejada é a formulação do QuadCut para problemas com função de energia que incluem termos de análise temporal, tais como formulações para análise de vídeo.

Um maior desafio é encontrado na extensão do QuadCut para aplicações que avaliam mais de uma imagem não alinhadas, como em pares de imagens estéreo ou pares de imagens para extração de fluxo ótico. A representação de ambas as imagens do par usando uma Quadtree impõe alinhamentos às partições resultantes das regras de subdivisão espacial da quadtree. Tais alinhamentos não apresentam necessariamente uma correspondência com a divisão maximal de similaridade no conteúdo da imagem. A extensão da QuadCut neste contexto pode se beneficiar das propostas publicadas em (Hon04, Ble07), as quais tratam do agrupamento de pixel em regiões por similaridade, previamente à construção do grafo, considerando planos de similaridade no interior das regiões produzidas. Tal consideração permite que os algoritmos classifiquem corretamente o interior de áreas sem textura – conhecido como problema da abertura ("aperture problem"), uma vez que a análise de movimento é avaliada para toda a região (previamente segmentada) como um todo.

A pesquisa apresentada nesta tese enfrentou uma mudança de desafios para a computação em GPU, ao longo do desenvolvimento dos algoritmos de visão computacional e processamento de imagens propostos.

O uso do poder de computação do hardware gráfico via codificação em linguagens de shaders, ou seja, via programação de etapas do pipeline gráfico, apresenta desafios tais como: requer um conhecimento gráfico significante, oferece um ambiente estranho de programação (com poucos recursos para análise de erros ou depuração) e exige formulações, muitas vezes, nada triviais. Outras dificuldades e limitações da programação genérica quando pensada sobre o pipeline gráfico incluem: as restrições dos mecanismos disponíveis para escrita indireta ("scatter"), os limites no número de entradas, saídas, constantes, registradores e instruções (limites que foram sendo melhorados a cada arquitetura); a impossibilidade de leitura imediata de resultados ("immediate output readback"); a inexistência de ordem de execução entre as computações em um fluxo de entidades; a falta de suporte ao tipo double; a inexistência de mecanismos de sincronização entre unidades de processamento computando uma determinada operação sobre elementos distintos. Um importante papel no desenvolvimento de aplicações genéricas em GPU é desempenhado pela comunidade de GPGPU que divulga padrões de programação paralela, para auxiliar o tratamento de tais desafios e formula abstrações das computações do pipeline gráfico que facilitam a compreensão da programação de "shaders"para programadores não

familiarizados aos procedimentos gráficos.

A programação genérica em GPU usando linguagens não gráficas (ex: CUDA), também apresenta desafios próprios tais como: a compreensão e gerenciamento de sua hierarquia de memória; a distribuição das operações aritméticas de forma a esconder a latência de memória; a ocupação e balanceamento dos multiprocessadores com repartições de blocos e threads apropriadas; o desenvolvimento de aplicações escaláveis com o número de processadores; entre outras.

De maneira geral, seja via programação de "shaders"ou via linguagens como CUDA, o desafio maior que permanece para o desenvolvimento de aplicações de visão computacional é expressá-las como computações paralelas à dados ("data-parallel computations"). Conforme apresentado ao longo da tese, os algoritmos passam a descrever computações de forma que um mesmo programa seja executado em paralelo a muitos elementos de dados.

Na medida em que a programação em GPU se torna mais flexível, ocorre uma convergência dos desafios encontrados para a implementação de tarefas de visão computacional aos desafios naturais da área de pesquisa em sistemas paralelos. Portanto, acreditamos que identificar quais são os elementos, a serem processados de maneira independente e paralela que compõem os fluxos de dados de uma determinada aplicação e descrever os algoritmos reformulando-os sob o ponto de vista de tais elementos se mantém como principal desafio à tarefa desempenhada nesta tese.