ESTUDO E INTEGRAÇÃO DA ARQUITETURA CUDA COM A PLATAFORMA OPENCL

Denise Alves da Costa (Aluna ICV/UFPI), Kelson Rômulo Teixeira Aires (Colaborador, Depto de Informática e Estatística - UFPI)Vinícius Ponte Machado (Orientador, idem)

Introdução

A área de Inteligência Computacional hoje requer cada vez mais sistemas de visão computacional com ótimo desempenho. O grande problema no entanto é o grande poder computacional que muitas das aplicações exigem. Se o hardware não tiver alto poder de processamento esses sistemas podem ter seus resultados prejudicados.

Foi na tentativa de sanar problemas como esses que a empresa NVIDIA desenvolveu a tecnologia CUDA (Computing Unified Device Architecture) [1]. Com objetivos parecidos a Khronos lançou no mercado uma plataforma de programação paralela chamada OpenCL (Open Computing Language) [2]. O objetivo deste trabalho foi realizar um estudo aprofundado nessas duas ferramentas e desenvolver algoritmos que integrassem ambas com trazendo uma nova proposta de desevolvimento de algoritmos de processamento de imagens.

Metodologia

Antes que se pudesse fazer qualquer atividade foi necessário realizar um estudo acerca de CUDA e OpenCL, a maneira de como as duas trabalham e como desenvolver algoritmos que pudessem desfrutar de suas vantagens. Para isso foram utilizados tutoriais desenvolvidos pelas próprias empresas desenvolvedoras das tecnologias em questão. Além desses materiais foram encontrados alguns outros que auxiliavam no entendimento das mesmas.

Com as tecnologias conhecidas foram desenvolvidos pequenos algoritmos, simples, mas que ajudavam no entendimento do uso adequado das ferramentas. Em seguida foram selecionados algoritmos mais complexos, já conhecidos no meio científico, para serem desenvolvidos utilizando a paralelização proposta pela CUDA com o auxílio do OpenCL. Os algoritmos foram selecionados conforme a relevância para o meio científico, respeitando os conhecimentos básicos do discente e que obviamente poderiam ser adequados a uma programação paralela.

Observando esses quesitos os principais algoritmos desenvolvidos foram: detecção de bordas com filtro Sobel [3] e determinação do fluxo óptico em seqüência de imagens. Tendo-se escolhido os algoritmos a serem desenvolvidos um estudo aprofundado acerca dos mesmos foi iniciado. Devendo-se fazer ainda a observação de que a segunda implementação citada dependia do sucesso da primeira.

Após o estudo e implementação dos algoritmos citados, os resultados foram analisados. E depois de análises e correções nos códigos desenvolvidos os programas mostraram desempenhos satisfatórios.

Resultados e discussões

Os resultados das implementações foram na verdade surgindo aos poucos, uma vez que durante toda a pesquisa não foi apenas uma implementação única desenvolvida, mas sim várias. À medida que cada etapa da pesquisa era alcançada e superada, pudemos ver os resultados da mesma. Dessa forma, os algoritmos puderam ser melhorados e até corrigidos ao longo da pesquisa.

Inicialmente os resultados não foram muitos satisfatórios, tanto em relação à saída dos programas, quanto ao próprio tempo de execução dos mesmos. Quanto à detecção de bordas, as imagens resultantes do algoritmo não apresentaram traços bem definidos do que realmente deveria ser exibido. Um exemplo é mostrado a seguir, onde a partir de um quadro de Romero Britto conseguiu-se inicialmente o seguinte resultado:

Figura 2 Imagem a ser utilizada no algoritmo Sobel

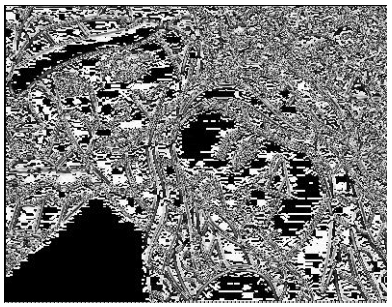

Figura 1 Resultado da primeira versão do algoritmo Sobel

Como pode ser claramente percebido, a imagem resultado não corresponde ao que de fato é esperado de um algoritmo de detecção de bordas. O problema residia no fato de que o resultado dos cálculos executados com relação a um pixel muitas vezes estava fora do intervalo válido de valores, que varia de 0 a 255, o que provocava uma falha nos resultados do algoritmo. Antes que se pudesse detectar o problema, outros meios foram procurados para sancioná-lo. Um deles foi desenvolver um algoritmo para retirar ruídos da imagem, ou seja, suavizar a imagem original antes de se calcular sua derivada. O processo melhorava o resultado, uma vez que diminuía o contraste na imagem, reduzindo assim “falsas bordas” que surgiam anteriormente. Os resultados melhoraram de fato, mas não era ainda o suficiente.

Para obter resultados ainda melhores, desenvolveu-se mais um algoritmo de auxílio, dessa vez para aplicar a binarização. Essa consiste da técnica de tornar uma imagem em escala de cinza em uma nova que possua apenas duas cores: branco e preto. Fazendo-se isso, conseguimos destacar as bordas verdadeiras que buscamos. E então conseguimos de fato exibir a derivada da imagem.

Porém, como mencionado anteriormente, uma falha fora detectada no algoritmo implementado. Após ser identifica, a mesma foi corrigida e o programa desenvolvido conseguiu mostrar resultados bastante satisfatórios. O algoritmo de detecção de bordas em OpenCL e CUDA estava finalmente solucionado.

Em seguida, com o auxílio do algoritmo citado acima, outro foi selecionado e desenvolvido. Trata-se do Fluxo Óptico, que consiste em detectar movimento aparente em uma seqüência de imagens [4]. Além do cálculo da derivada de imagens, utilizou-se ainda outro que consiste em calcular a diferença entre imagens.

Quanto aos resultados com relação ao ganho tempo, o qual foi a grande motivação do desenvolvimento desse projeto, os resultados são mostrados a seguir. O tempo é dado em milisegundos.

Tabela 1 Tempo de processamento do algoritmo de detecção de bordas

Resolução da imagem	208x208	576x432	640x480	700x558	800x600	1280x800	2816x2112
Tempo em CPU	9	41	78	61	70	149	813
Tempo em GPU	4	18	31	29	34	74	430
Temp. em CPU/ Temp. em GPU	2,25	2,28	2,52	2,1	2,06	2,01	1,89

Tabela 2 Tempo de processamento do cálculo do fluxo óptico

Resolução da imagem	193x140	576x432	640x480	969x500	1128x960	1280x800	2816x2112
Tempo em CPU	15	131	164	266	640	544	3111
Tempo em GPU	11	76	93	141	342	295	1676
Temp CPU/Temp GPU	1,36	1,72	1,76	1,89	1,87	1,84	1,86

Conclusões

Os principais algoritmos aqui desenvolvidos foram o de detecção de bordas e fluxo óptico. Com o objetivo de auxiliá-los, outros ainda foram desenvolvidos, como por exemplo: suavização, binarização e diferença de imagens.

Como pode ser notado nas tabelas da sessão anterior, o ganho de tempo de processamento é notório. O objetivo do projeto de pesquisa foi alcançado, o qual se baseou num estudo aprofundado em OpenCL e CUDA e o desenvolvimento de algoritmos utilizando tais tecnologias, e com isso ganhar tempo no processamento de imagens.

Referências Bibliográficas

[1] NVIDIA, CUDA Programming Guide Version 3.0 2010, NVIDIA Corporation: Santa Clara Califórnia.

[2] Khronos. OpenCL - The open standard for parallel programming of heterogeneous systems. Retrieved August 15, 2011, from http://www.khronos.org/opencl

[3] Gonzalez, Rafael C. e Richard E. Woods, Digital Image Processing, 3rd ed, Prentice Hall, 2007.

[4] Park, Seung and Ponce, Sean et al, Low-cost, high-speed computer vision using NVIDIA's CUDA architecture, 37 th IEEE Applied Imagery Pattern Recognition Workshop, Washington, DC, USA, 2008.

Palavras-chave: Processamento de imagens. CUDA. OpenCL.