PARALELIZAÇÃO DE ESQUELETIZAÇÃO DE IMAGENS DE FUNDO DE RETINA NA ARQUITETURA CUDA

Karin S. Komati[1], Flavio S. L. de Souza[1], Juliana A. Guimarães[1], Jefferson O. Andrade[1]
[1] Instituto Federal do Espírito Santo - IFES
kkomati@ifes.edu.br, flaviolamas@ifes.edu.br, juju.guimaraes@gmail.com, joandrade@ifes.edu.br

Resumo
Este trabalho apresenta uma análise comparativa do tempo de resposta de esqueletização de imagens, usando as diretrizes do algoritmo Zhang-Suen, desenvolvido sob duas formas: sequencial mono-processada e paralela multi-processada usando uma unidade de processamento gráfico. A plataforma de computação paralela escolhida foi o CUDA. A aplicação é voltada para imagens de fundo de retina, cuja extração de características dos vasos sanguíneos auxiliará diagnósticos médicos e, portanto, o tempo de resposta do sistema é fundamental. Testes realizados em uma base de dados pública de imagens de retina, DRIVE, mostraram que a versão paralela foi, em média, mais de 31 vezes mais rápida do que a versão sem paralelismo.

Palavras-chave: Algoritmo de esqueletização. Zhang-Suen. Imagens digitais de fundo de retina. CUDA. DRIVE.

Abstract
This work presents a comparative analysis of the response time for the skeletonization of images, using two versions of the Zhang-Suen algorithm: a sequential mono-processed version and an parallel multi-processed version using the graphics processing unit. The parallel computing platform chosen was CUDA. The skeletonization applications developed is aimed towards processing retinal images, whose characteristics are extracted of blood vessels to assist medical diagnosis, and thus the response time of the system is paramount. Tests were performed on the DRIVE public retinal images database, and showed that the parallel version of the algorithm was, on average, more than 31 times faster than the sequential version.

Keywords: *Skeletonization algorithm. Zhang-Suen. Digital retinal images. CUDA. DRIVE.*

1. Introdução

A visão é um dos sentidos do ser humano, e é por meio desse sentido que temos a capacidade de enxergar tudo à nossa volta. Os olhos são os órgãos responsáveis pelo sentido da visão, a anatomia do olho humano é ilustrada na Figura 1. Apresenta na parte anterior do olho, uma área transparente com maior curvatura, chamada de córnea. A coroide é uma película dotada de vasos sanguíneos e melanina que tem a função de nutrir e absorver a luz que chega à retina. Na parte anterior da coroide localiza-se a íris, estrutura muscular de cor variável. Na íris há um orifício central que chamamos de pupila. É por esse orifício que há a entrada da luz no globo ocular.

O cristalino se situa atrás da íris e é uma lente biconvexa que orienta a passagem de luz até a retina. Está cercado por fluidos na parte anterior e posterior. Na parte anterior, há uma câmara preenchida pelo humor aquoso, enquanto que na parte posterior, há uma câmara preenchida com um líquido viscoso e transparente chamado de humor vítreo.

A retina é uma membrana mais interna e se encontra abaixo da coroide, que é composta por células visuais que nos permitem reconhecer luz e cores. A retina é a responsável pela formação de imagens, é como uma tela onde se projeta a luz: retém as imagens e as traduz para o cérebro através de impulsos elétricos enviados pelo nervo óptico. A fundoscopia é uma técnica de observação do fundo do olho para análise da retina e outras estruturas internas. Uma imagem de fundo de retina é apresentada na Figura 2, onde é possível se ver as suas artérias e veias.

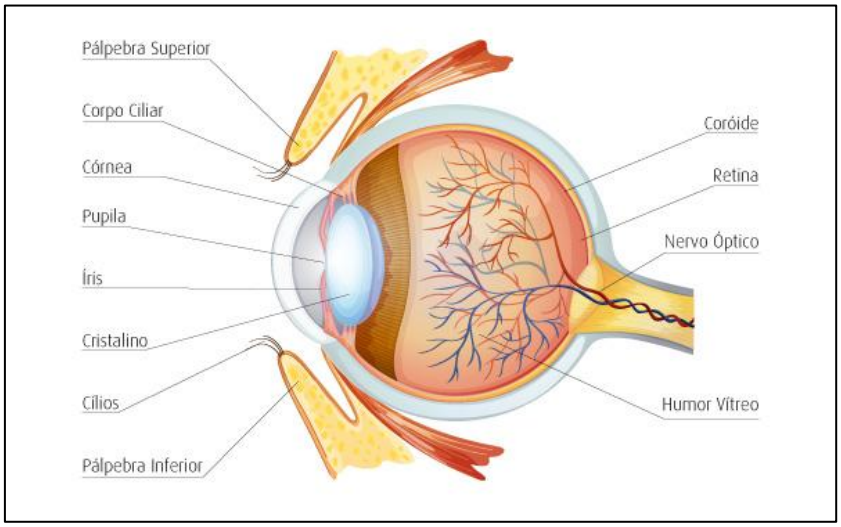


Figura 1 – Anatomia do olho humano. [imagem traduzida]. (Fonte: 123RF, 2013)

Figura 2 – Imagem de fundo de retina (Fonte: STALL et. al., 2004)

Imagens de fundo de retina permitem não apenas a avaliação de alterações oculares, mas também a detecção de doenças sistêmicas, pois estas podem causar retinopatia (formas de lesões não inflamatórias da retina ocular). Assim, a morfologia dos vasos sanguíneos pode indicar a presença de diversas doenças, tais como hipertensão e diabete, e também é útil para determinar seu avanço (ZEPEDA-ROMERO et. al., 2011). A aquisição de imagens de fundo de retina é uma tarefa praticamente não invasiva, necessita-se apenas de algumas gotas de medicamento a serem aplicadas aos olhos para causar a dilatação da pupila (THYPARAMPIL et. al., 2010).

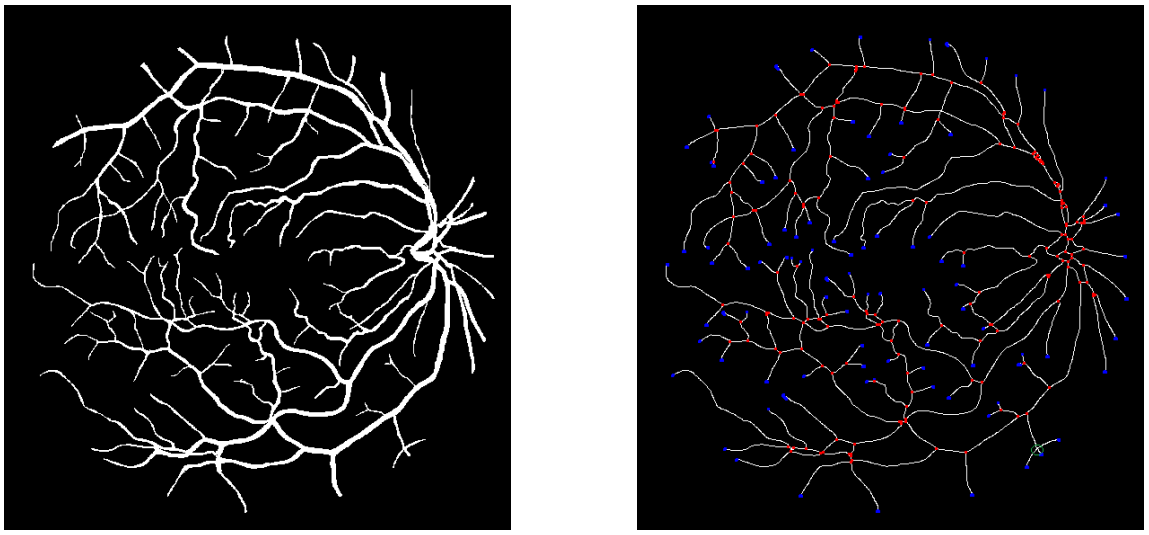
Estas ramificações sanguíneas podem ser estudadas computacionalmente. Com técnicas de processamento digital de imagens (PDI), é possível segmentar as veias e artérias, e conhecer seu comprimento e diâmetro, dentre outras medidas. Com a extração da "árvore sanguínea", é possível se calcular a tortuosidade, simetria, ângulo de bifurcação, e conectividade, por segmento. Com estes dados, um médico dispõe de indicadores que o auxiliarão nos diagnósticos.

Para a extração das informações de cada ramificação dos vasos sanguíneos, a imagem colorida da retina passa por várias etapas. A primeira etapa é a segmentação, separando quais pontos da imagem pertencem ou não pertercem aos vasos sanguíneos, gerando uma imagem binária (preta e branca) como mostrada na Figura 3(a) a partir da imagem da Figura 2. A base de dados DRIVE (STALL et. al., 2004) possui uma série de imagens binárias de vasos sanguíneos que foram utilizadas neste trabalho.

A próxima etapa é o processo de esqueletização que afina as linhas representativas de vasos sanguíneos na imagem até que sobrem apenas linhas finas, tal qual um esqueleto. Com base nesse esqueleto, é possível extrair os pontos de bifurcação (os pontos na "árvore sanguínea" onde

ocorre a separação de um vaso em dois ou mais ramos), os pontos finais, identificar cada segmento, e extrair suas características. A Figura 3(b) ilustra o resultado do algoritmo de esqueletização Zhang-Suen (ZHANG; SUEN, 1984) aplicada à imagem da Figura 3(a), com os pontos de bifurcação marcados em vermelho e os pontos finais marcados em azul. Esta imagem foi obtida como resultado do trabalho de Guimarães, Souza e Komati (2012).

Após a identificação dos pontos de controle (pontos de bifurcações e pontos finais), deve-se fazer o cômputo das características de cada ramo, tais como comprimento e diâmetro. Esta etapa não é tratada neste trabalho.



(a) (b)

Figura 3 – (a) Imagem segmentada da imagem de fundo de retina (STALL et. al., 2004) (b) Imagem esqueletizada e com pontos de bifurcação e pontos final marcados em cores

Embora trabalho citado acima já tenha chegado ao resultado da Figura 3(b) a partir da image da Figura 3(a), os usuários reclamavam da demora no tempo de processamento. Em média, o sistema possuía tempo de resposta de 95 ms. Importante salientar que em uma possível aplicação deste algoritmo para o processamento de vídeo que tenha 30 quadros por segundo, onde cada quadro ocupa 33 ms, o tempo de resposta de 95 ms inviabilizaria o processamento de vídeo.

Assim, neste trabalho, o objetivo é avaliar o tempo de execução do algoritmo de esqueletização de Zhang-Suen de duas formas: (a) com uma implementação sequencial mono-processada; e (b) com uma implementação paralela multi-processada utilizando uma unidade de processamento gráfica (GPU – *Graphic Processor Units*). Para tanto, utilizou-se o *Compute Unified Device Architecture* - CUDA (NVIDIA, 2012a), uma plataforma de computação paralela criada pela NVIDIA, que permite aumento do desempenho computacional ao aproveitar a capacidade de processamento da GPU.

O algoritmo de Zhang-Suen já foi concebido para ser tratado tanto sequencial quanto paralelamente. A contribuição deste trabalho é a implementação de sua versão paralela na plataforma CUDA, bem como comparar quantitativamente a melhoria no tempo de resposta entre as versões sequencial e paralela em uma configuração de *hardware* típica atual.

Este artigo está dividido da seguinte forma: na próxima seção apresenta-se a revisão teórica apresentando o algoritmo de esqueletização Zhang-Suen e a plataforma de computação paralela CUDA; na terceira seção citam-se alguns trabalhos relacionados; na quarta seção são apresentadas a plataforma de desenvolvimento e o modelo de programação; na quinta seção descrevem-se os experimentos e resultados; e por fim, na última seção, as considerações finais desse artigo.

2. Revisão Teórica

Há vários algoritmos de esqueletização, tais como Stentiford, Holt e Morfologia Matemática (via transformada *Hit-and-Miss*). Utilizando os resultados de Corrêa e Festa (2005), escolheu-se o algoritmo de Zhang-Suen, pois não altera a morfologia da árvore sanguínea. Embora, o algoritmo de Zhang-Suen apresente como principal desvantagem um resultado chamado de *staircase*, ou efeito escada, que é esteticamente desagradável.

Esta seção se divide em quatro partes: conceitos sobre topologia digital de imagens (que apresenta conceitos que serão úteis para a compreensão do algoritmo de esqueletização), o algoritmo de esqueletização Zhang-Suen sequencial e paralelo e a plataforma de computação paralela CUDA. Na seção seguinte se explica como os conceitos serão combinados para o desenvolvimento da solução, como o algoritmo paralelo de Zhang-Suen foi implementado em CUDA.

2.1. Topologia Digital

Apresentam-se alguns conceitos sobre topologia digital de imagens, (isto é, a propriedade entre pixels), que serão necessários para a compreenção do algoritmo de esqueletização imagens (GONZALEZ; WOODS, 2008). Uma imagem binária é uma matriz bidimensional, na qual cada pixel tem o valor 0 (cor preta) ou valor 1 (cor branca). Considere a nomenclatura de pixels de P1 a P9, conforme Figura 4.

P_3	P_2	P_9
P_4	P_1	P_8
P_5	P_6	P_7

Figura 4. Pixel de uma imagem e sua vizinhança.

Se considerarmos apenas os vizinhos na vertical (P2 e P6) e na horizontal (P4 e P8) de um pixel, P1, então se têm a "Vizinhança de 4". A "Vizinhança de 8" é quando consideramos todos os vizinhos de P1, isto é, os pixels de P2 à P9. Podem-se considerar vizinhas mais distantes, além da matriz 3x3.

A conectividade é um conceito usado para estabelecer fronteiras de objetos e regiões em uma imagem, onde dois pixels são conectados se são vizinhos e suas cores satisfazem a um critério especificado de similaridade. Para o algoritmo de Zhang-Suen, consideramos conectividade como pixels em Vizinhança de 8 que tenham o mesmo valor.

2.2. Algoritmo de Zhang-Suen

Um algoritmo de esqueletização reduz os vasos sanguíneos da imagem a uma linha fina. A esqueletização elimina distorções no contorno, e mantém as propriedades geométricas e topológicas da imagem, permitindo a obtenção de propriedades críticas como pontos extremos e conexões. Utilizou-se o método Zhang-Suen, cuja ideia é decidir se um determinado pixel será eliminado analisando o valor dos seus oito vizinhos (Vizinhança de 8).

Há duas diretivas para decidir se o pixel deve ou não ser removido:

- a primeira diretiva diz que o pixel somente pode ser apagado se o número de conectividade do mesmo for igual a um. Isto significa que o pixel é conectado somente a uma única região. Se um pixel possuir o número de conectividade igual a dois então, duas regiões conectadas poderão se separar e isto não pode ocorrer;

- A segunda diretiva é que o pixel somente pode ser apagado se este tiver mais de um e menos de sete vizinhos. Esta regra assegura que os pixels resultantes foram retirados sucessivamente das bordas da região da imagem, e não de suas partes internas.

As duas diretivas do algoritmo de Zhang-Suen são analisadas em duas sub-iterações, onde uma sub-iteração exclui os pixels de contorno norte e oeste, e a outra sub-iteração exclui os restantes, sul e leste. É importante ressaltar que os pixels só devem ser eliminados no final de cada sub-iteração. Se no final da segunda sub-iteração não existirem pixels para serem eliminados, então a esqueletização está completa e o programa pára.

Em cada sub-iteração existem quatro regras que devem ser aplicadas. Se, e somente se, as quatro forem satisfeitas, o pixel poderá ser eliminado. Estas quatro regras garantem as duas diretivas citadas e são:

1) $2 \leq NV(P1) \leq 6$;
   onde NV(P1) é o número de vizinhos com valores diferente de 0. Significa que:
   - um pixel sem vizinhos não será apagado, pois é um ponto isolado,
   - um pixel com apenas um vizinho não será apagado, pois é um ponto terminal,
   - um pixel com 7 vizinhos não será apagado, pois está localizado numa concavidade e,
   - um pixel com 8 vizinhos não será apagado, pois está na parte interna de uma região.
2) NT(P1) = 1;
   onde NT(P1) é o número de transições do valor 0 para o valor 1 dos vizinhos de P1, na sequência P2, P3, ..., P9. O termo NT significa a conectividade, isto é, é a definição que se enconta na primeira diretiva.
3) P2 x P6 x P8 = 0.
   Ao menos um dos vizinhos: o acima, à direita ou abaixo são fundo.
4) P4 x P6 x P8 = 0
   Ao menos um dos vizinhos: à esquerda, à direita ou abaixo são fundo.

No final desta sub-iteração os pixels marcados são eliminados. A próxima sub-iteração as mesmas regras, exceto para os passos 3 e 4, logo:

3) P2 x P4 x P8 = 0.
   Ao menos um dos vizinhos: o acima, à direita ou à esquerda são fundo.
4) P2 x P4 x P6 = 0
   Ao menos um dos vizinhos: à esquerda, acima ou abaixo são fundo.

Novamente, todos os pixels marcados são eliminados.

2.3. Algoritmo de Zhang-Suen Paralelo

Muitos dos algoritmos de esqueletização não podem ser paralelizados, pois para que um pixel seja processado, leva-se em conta o resultado do passo anterior e também do passo atual. Já para um algoritmo poder ser paralelizado, a decisão de remover ou não um pixel na iteração atual é baseada somente no resultado da iteração (passo) anterior.

A vantagem do algoritmo de Zhang-Suen é que pode ser paralelizado, pois é um algoritmo de esqueletização dividido em duas sub-iterações independentes, por preservar a conectividade dos pixels. Caso contrário, um simples exemplo de um retângulo horizontal que possui dois pixels de largura pode vir a desaparecer durante o processo de afinamento.

Como a decisão de remover ou não um pixel na sub-iteração atual é baseada somente no resultado da sub-iteração anterior, todos os pixels podem ser analisados independentemente e de uma forma paralela a cada sub-iteração. Uma vez que, a partir da primeira sub-iteração, todas as

outras são aplicadas sobre o seu resultado, se possuirmos uma CPU por pixel pode-se rodar todas as sub-iterações subsequentes de uma única vez.

É importante ressaltar a importância de se atualizar a imagem apenas no final de cada sub-iteração. A eliminação dos pixels no instante de seu processamento altera o resultado do algoritmo, pois é importante que a imagem não se altere no decorrer de cada sub-iteração.

2.4. CUDA

A arquitetura CUDA permite que o programador desenvolva algoritmos que executam parte de suas instruções no processador da placa de vídeo e parte na CPU, tornando, assim, a execução paralela. CUDA utiliza uma linguagem de alto nível que é uma extensão da linguagem de programação C padrão, adicionando uma biblioteca específica com funções que auxiliam o programador (DUTRA; VARINI; CANAL, 2012).

A arquitetura CUDA é constituída por *Streaming Multiprocessors* (SM) chamados de Multiprocessadores. Cada multiprocessador é formado por um grupo de núcleos de processamento (*Scalar Processors* - SP), em outras palavras, um número **n** de núcleos encapsulados forma um multiprocessador. Cada multiprocessador executa instruções de forma independente e paralela em relação aos demais. Os multiprocessadores possuem uma arquitetura chamada de SIMT (*Single Instruction, Multiple Thread*), onde todos os núcleos de um mesmo grupo executam a mesma instrução (*kernel*) de forma paralela.

Thread é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente. Uma *thread* permite, por exemplo, que o usuário de um programa utilize uma funcionalidade do ambiente enquanto outras linhas de execução realizam outros cálculos e operações. Em hardwares equipados com uma única CPU, as *threads* são processadas de forma aparentemente simultânea por *time sharing*, pois a mudança entre uma *thread* e outra é feita de forma tão rápida que para o utilizador isso parece acontecer simultaneamente. Em *hardwares* com múltiplas CPUs ou multi-cores, as *threads* são executadas realmente de forma simultânea.

CUDA segue o conceito básico de blocos. Cada bloco é formado por um número pré-definido de *threads* que são também organizadas em forma de matriz. Os blocos apresentam uma estrutura interna, onde cada posição contém uma *thread* diferente. A dimensão dos blocos é definida pelo programador na criação de um novo *kernel*. Cada bloco é mapeado a um multiprocessador de forma automática pelo dispositivo, o número de blocos é independente do número de multiprocessadores existentes no dispositivo.

Na Figura 5, é ilustrado um modelo de GPU que contém um número **n** de multiprocessadores (SM) cada um deles contendo 8 núcleos de processamento (SP). Se fossem criados blocos de 32 threads cada um, elas seriam divididas em pequenos grupos de 4 threads, cada grupo seria atribuído a um núcleo diferente. O número de *threads* por bloco é independente do número de núcleos (SP), mas cada modelo de GPU possui um número máximo suportado (PILLA, 2009).

O paradigma simplifica o processamento de fluxo de software e hardware paralelo. Dado um conjunto de dados (*stream*), uma série de operações de funções (*kernel*) é aplicada a cada elemento na *stream*. O *kernel* pode ser definido como a parte paralela do código onde as *threads* são mapeadas aos multiprocessadores do dispositivo (GPU).

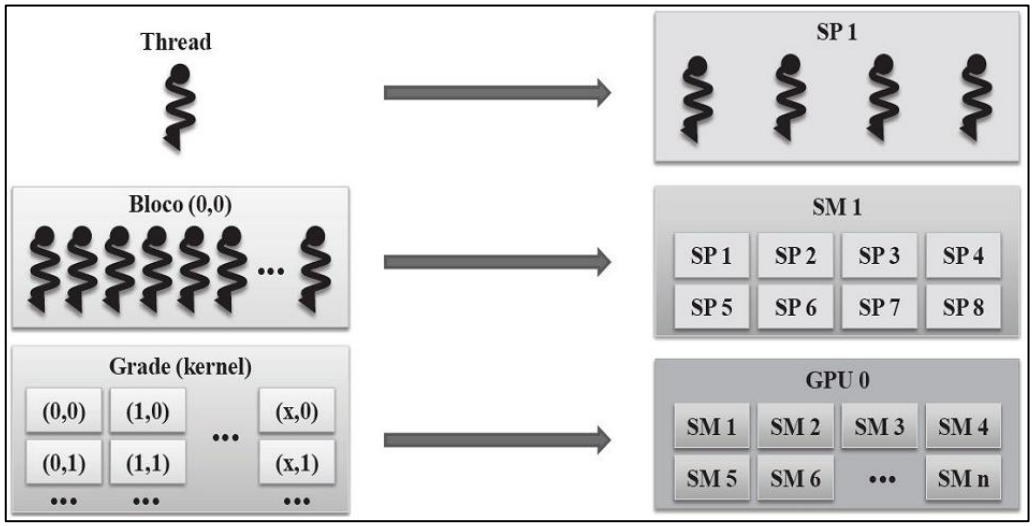


Figura 5. Mapeamento das threads para os núcleos de processamento (PILLA, 2009)

3. Trabalhos relacionados

Algoritmos de processamento de imagens em computação paralela foram desenvolvidos em vários trabalhos. Um deles, o trabalho de Nugteren, Corporaal e Mesman (2011) faz a análise de desempenho de doze algoritmos diferentes da área de processamento de imagens utilizando CUDA, no entanto, o algoritmo de esqueletização não fez parte deste conjunto de doze algoritmos. A comparação dos resultados se baseou nos fatores de como a memória compartilhada e a largura de banda afetaram o desempenho em termos de tempo de execução dos sistemas.

Já o trabalho de (GULO et. al., 2012) também implementa um algoritmo de processamento de imagens em CUDA, no entanto, o algoritmo desenvolvido foi o método de suavização de imagens. O ganho em tempo de execução foi de 3,5 vezes melhor para a versão paralela comparada com a versão sequencial.

O trabalho de (PALOMERA-PEREZ et. al., 2010) implementa um algoritmo de segmentação em imagens de retina, e mostra que a versão usando computação paralela foi de 8 a 10 vezes mais rápida que a versão sequencial.

Em todos os trabalhos, o resultado da versão paralela teve um desempenho superior. Embora, todas as aplicações sejam na área de processamento de imagens, nenhum deles tratou do algoritmo de esqueletização.

4. Solução Proposta

O sistema foi desenvolvido em linguagem C, utilizando a biblioteca OpenCV (ITSEEZ, 2012), que é voltada para o desenvolvimento na área de visão computacional sob licença BSD. Geraram-se dois sistemas: um apenas com o OpenCV e outro com o OpenCV compilado com CUDA.

A placa de vídeo utilizada é a GeForce GTS 450 (NVIDIA, 2012b), que possui: 4 *Streaming Multiprocessors* (SM), com 48 núcleos para cada SM, totalizando 192 núcleos ou SP (4x48=192).

O modelo de programação adotado consistiu em criar uma *thread* para cada pixel. Criou-se um kernel com 640 blocos, e cada bloco com 512 *threads*, exatamente para maximizar a utilização de cada SM, dado que a placa de vídeo utilizada, possui limite máximo de gerência de 1536 *threads* por SM.

Cada bloco é alocado a um multiprocessador da GPU, que pode executar um ou mais blocos simultaneamente. A placa de vídeo usada possui limites de: 1024 threads por bloco e 1536 *threads* por multiprocessador.

Ao configurar o valor de 512 *threads* por bloco, cada multiprocessador passa a trabalhar em sua carga máxima de 1536 *threads*, pois passa a ter um gerenciamento de 3 blocos inteiros por cada multiprocessador. Caso fosse configurado o valor de 1024 *threads* para bloco ao invés das 512 *threads*, estaríamos deixando de usar 512 *threads* do limite máximo de cada multiprocessador, gerando um desperdício de 2048 *threads* que poderiam ser usadas para processar informação, ou 2048 pixels.

O fluxo de processamento seguiu o tipicamente usado, ilustrado na Figura 6:

1. O passo 1 (*Copy processing data*) é feito pelo função "**cudaMemcpyToArray(...,cudaMemcpyHostToDevice)**", onde se passa a imagem inteira da memória principal (*Host*) para a memória da GPU (*Device*);
2. O passo 2 (*Instruct the processing*) é o conjunto de comandos que definem os parâmetros a serem passados à função a ser paralelizada, bem como as variáveis que definem quantidade de blocos da grade e o tamanho de cada bloco;
3. O passo 3 (*Execute parallel in each core*) é feito pela chamada "**kernel <<< noOfBlock, blockSize >>> (par,..)**". Onde a sintaxe entre os símbolos <<< e >>> indicam a quantidade de blocos da grade e o tamanho de cada bloco. Os dois parâmetros são inseridos entre o nome da função e a sua lista de parâmetros. No caso, os parâmetros foram 640 e 512, os valores são fixos, pois todas as imagens possuíam o mesmo tamanho;
4. O passo 4 (*Copy the result*) é feito pelo função "**cudaMemcpyToArray(...,cudaMemcpyDeviceToHost)**", onde se passa a imagem inteira da memória da GPU (*Device*) para a memória principal (*Host*);

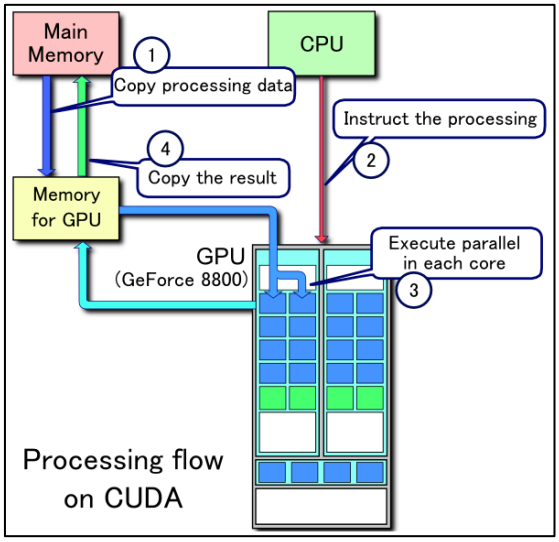


Figura 6. Fluxo de processamento em CUDA (Fonte: Wikimedia Commons, 2013).

5. Experimentos e Resultados

O computador de testes possui processador Intel core i7-870 2.93GHz, com placa-mãe Asustek p7H55-M BR, 8Gb de memória RAM com sistema operacional Windows 7 64bits sp1. Relembrando que a placa de vídeo utilizada é a GeForce GTS 450 (NVIDIA, 2012b).

Para testar nossa proposta, foram utilizadas todas as imagens da base de dados DRIVE (STAAL et. al, 2004). Esta base de dados possui 40 imagens de fundo de retina, com resultados de segmentações feitos por especialistas e segmentações automáticas executadas por cinco

diferentes sistemas. Utilizamos como entrada, as imagens feitas pelo especialista 1. O algoritmo de esqueletização manteve a estrutura geográfica e topologia da imagem em todos os casos.

A Tabela 1 mostra as medidas de tempo. Nesta tabela há uma coluna com os tempos do sistema "Sem CUDA", "Com CUDA" e a relação entre as duas medidas (Taxa = tempo Sem CUDA/ tempo Com CUDA). A primeira linha apresenta a média dos tempos obtidos nas 40 imagens. Observa-se que em média, o tempo sem CUDA é em torno de 31 vezes maior que o tempo com CUDA. Assim, verificamos que o uso de GPU realmente melhora em muito o desempenho do sistema em termos de tempo de resposta.

	Sem CUDA	*Com CUDA*	*Taxa*
Média de tempo de exeucação das 40 imagens	94,73 ms	3,07 ms	31,57
Caso de melhor desempenho	86,3 ms	1,89 ms	45,44
Caso de pior desempenho	89,4 ms	4,32 ms	20,70
Caso de maior tempo de execução	128,3 ms	5,02 ms	25,54

Tabela 1 – Medidas de tempos

Na segunda linha da Tabela 1, apresentam-se os dados de melhor desempenho, onde a taxa é de mais de 45 vezes, e a próxima linha são os dados para o caso de pior desempenho, onde a taxa é em torno de 20 vezes. Mesmo no pior caso, o desempenho diferencial é considerável.

A execução que demandou o maior tempo de execução em ambos os sistemas, é mostrada na Figura 7, na qual é possível se notar que a quantidade de pixels a serem analisados (em branco) é maior que a da Figura 3(a). No sistema sem CUDA, o sistema demorou mais de 128 ms para responder e com CUDA apenas 5 ms.

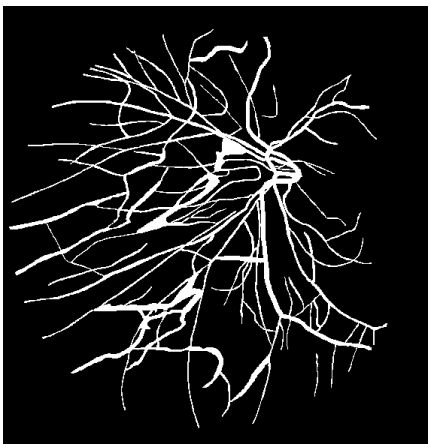


Figura 7 – Imagem segmentada de maior tempo de execução (STALL et. al., 2004)

6. Considerações Finais

O objetivo deste trabalho foi analisar quantitativamente o ganho no tempo de respota do algoritmo de esqueletização Zhag-Suen com o uso da plataforma de processamento paralelo CUDA sobre a sua versão sequencial mono-processada. O resultado foi que, em média, o uso da plataforma CUDA melhorou o tempo de execução do algoritmo em mais de 31 vezes quando comparado ao tempo de execução sem CUDA.

A esqueletização é apenas um passo dentre vários passos sequenciais para o desenvolvimento de um sistema completo de auxílio ao diagnóstico médico: segmentação da imagem, binarização, esqueletização, identificação de pontos de controle e cômputo das várias características de cada ramo da árvore sanguínea. Assim, a soma do tempo total de processamento dos vários processos deve ser otimizada para que o trabalho do usuário não seja prejudicado com o tempo de resposta do sistema.

Como trabalhos futuros, planeja-se:

- alterar a estratégia de paralelização, tratando a questão da concorrência. O objetivo é dividir a imagem em blocos de tamanhos diferentes, começando de blocos menores e crescer gradativamente, para avaliar qual é a divisão de melhor performance;
- explorar a capacidade da memória compartilhada (*shared memory*) do CUDA. A memória compartilhada rápida pode ser compartilhada entre as várias *threads*, podendo ser usada como um cache de usuário e melhorar mais ainda o desempenho e;
- avaliar outros algoritmos de processamento de imagens quanto ao tempo de resposta das versões com CUDA e sem CUDA.

Referências

123RF. **Illustration of the human eye anatomy**. Disponível em: http://www.123rf.com/photo_20185366_illustration-of-the-human-eye-anatomy.html. Acesso em: junho 2013.

CORRÊA, F. P.; FESTA, L. M. **Avaliação de Técnicas para Afinamento de Imagens Digitais**. Trabalho de Conclusão de Curso de Bacharelado em Ciência da Computação da Universidade Federal do Paraná, 2005.

DUTRA, E. R. F., VARINI, A. L., CANAL, A. P. **Paralelização de aplicações na arquitetura cuda: um estudo sobre vetores**. In: XVI SIMPÓSIO DE ENSINO, PESQUISA E EXTENSÃO (SEPE 2012), 2012. Santa Maria/RS.

ITSEEZ. **OpenCV**. Disponível em: http://opencv.org/. Acesso em: out. 2012.

GONZALEZ, R. C., WOODS, R. E. **Digital image processing**. 3ª ed. Prentice Hall, Upper Saddle River, NJ, 2008. 716p.

GUIMARÃES, J. A.; SOUZA, F. S. L.; KOMATI, K. S. **Identificação de Pontos de Bifurcação em Vasos Sanguíneos de Imagens de Retina Usando Algoritmo de Esqueletização**. In: III ERI-MT 2012 Escola Regional de Informática do SBC (Regional Mato Grosso), 2012, Rondonópolis. Anais do III ERI-MT 2012 Escola Regional de Informática do SBC (Regional Mato Grosso), 2012.

GULO, C. A. S. J., ARRUDA, H. F. de A., SEMENTILLE, A. C. , ARAUJO, A. F., TAVARES, J. M. R. S. **Método de Suavização de Imagem baseado num Modelo Variacional Paralelizado em Arquitetura CUDA**. In: XXXII Congresso da Sociedade Brasileira de Computação, GPU Computing Developer Forum (CSBC 2012), 2012. Curitiba/PR.

NUGTEREN, C., CORPORAAL, H., MESMAN, B., **Skeleton-based automatic parallelization of image processing algorithms for GPUs**. In: XI International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation, (ICSAMOS 2011), 2011, Samos, Greece. p. 25-32.

NVIDIA. **GeForce GTS 450**. Disponível em: http://www.nvidia.com.br/object/product-geforce-gts-450-br.html. Acesso em: out. 2012.

NVIDIA. **Cuda**. Disponível em: http://www.nvidia.com.br/object/cuda_home_new_br.html. Acesso em: out. 2012.

PALOMERA-PEREZ, M. A., MARTINEZ-PEREZ, M. E., BENITEZ-PEREZ, H., ORTEGA-ARJONA, J. L.. Parallel multiscale feature extraction and region growing: application in retinal blood vessel detection. **IEEE Transactions on Information Technology in Biomedicine**. 14(2):500-506, 2010.

PILLA, L. L. **Análise de Desempenho da Arquitetura CUDA Utilizando os NAS Parallel**. Trabalho de Conclusão (Graduação em Ciência da Computação). Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre. 2009.

STAAL, J. J.; ABRAMOFF, M. D.; NIEMEIJER, M.; VIERGEVER, M. A.; VAN GINNEKEN, B. Ridge based vessel segmentation in color images of the retina. **IEEE Transactions on Medical Imaging**, vol. 23, pp. 501-509. 2004.

THYPARAMPIL, P. J., PARK, Y., MARTINEZ-PEREZ, M. E., LEE, T. C., WEISSGOLD, D. J., BERROCAL, A. M., CHAN, R. V. P., FLYNN, J. T., CHIANG, M. F. Plus Disease in Retinopathy of Prematurity: Quantitative Analysis of Vascular Change. **American Journal of Ophthalmology.** 150(4):468-475.e2, 2010.

WIKIMEDIA COMMONS. **File:CUDA processing flow (En).PNG**. Disponível em: http://en.wikipedia.org/wiki/File:CUDA_processing_flow_(En).PNG. Acesso em: junho 2013.

ZEPEDA-ROMERO, L. C., MARTINEZ-PEREZ, M. E., RUIZ-VELASCO, S., RAMIREZ-ORTIZ, M. A.; GUTIERREZ-PADILLA, J. A. Temporary morphological changes in plus disease induced during contact digital imaging. **Eye**. 25(10):1337-1340, 2011.

ZHANG T.Y; SUEN C.Y. A Fast Parallel Algorithm for Thinning Digital Patterns, **Communications of ACM**, vol 27, nº 3, pp 236-239, 1984.