COMPARAÇÃO DE MÉTODOS DE COMPUTAÇÃO PARALELA PARA O PROCESSAMENTO DE IMAGENS DE PUPILOMETRIA

M. S. Oliveira[1], N. C. P. Pereira[2], J. K. S. Souza[1], G. E. Ginani[3], S. Pompéia[3], C. J. Tierra-Criollo[1,4] e D. B. Melges[1]

[1] Programa de Pós-Graduação em Engenharia Elétrica - UFMG, Belo Horizonte, Brasil
[2] Graduação em Engenharia de Controle e Automação, Belo Horizonte, Brasil
[3] Departamento de Psicobiologia - UNIFESP, São Paulo, Brasil
[4] Programa de Engenharia Biomédica - UFRJ, Rio de Janeiro, Brasil
e-mail: marceloliveira@ufmg.br

Resumo: Este trabalho tem por objetivo comparar o desempenho de três métodos para o processamento de vídeos de pupilometria, visando análise *online*, quais sejam: i) sequencial; ii) paralelo usando a biblioteca OpenMP e os diversos núcleos do processador; iii) paralelo usando a plataforma CUDA (*Compute Unified Device Architecture*) em uma placa gráfica. Para tanto foram empregados vídeos com duração de 10 minutos, resolução de 320 x 240, adquiridos a 120 Hz. Os três métodos foram comparados usando-se teste de Kruskal-Wallis aplicado ao parâmetro tempo de execução. Além disso, o parâmetro *Speed-up* foi usado para comparar os algoritmos paralelizados por meio do teste Wilcoxon Rank-Sum. Os tempos de execução foram estatisticamente diferentes para os três métodos, indicando vantagem do CUDA sobre OpenMP e deste sobre o sequencial. Dentre os algoritmos paralelos, o *Speed-Up* para o CUDA também foi estatisticamente superior ao do OpenMP. Com base nos resultados, ambos os métodos paralelos são adequados para agilizar o processamento de vídeos de pupilometria, embora haja larga vantagem para o processamento com CUDA.
Palavras-chave: Pupilometria, CUDA, OpenMP, GPU.

Abstract: *This research aims at comparing the performance of three methods for pupillometry video processing, in order to provide online video analysis, which are: in) sequential; ii) a parallel algorithm using OpenMP library in a multicore processor; iii) a parallel algorithm using CUDA (Compute Unified Device Architecture) platform in a graphic card. For this purpose, ten-minutes videos recorded in a 320 x 240 resolution and at 120 Hz were used. The three methods were compared using the Kruskal-Wallis statistical test, applied to the execution time parameter. Furthermore, the Speed-up parameter was used to compare the parallel algorithms by using the Wilcoxon Rank-Sum test. The execution times were statistically different for all three methods, indicating advantage of CUDA over OpenMP and from both over the sequential one. Among the parallel algorithms, both methods are appropriate to improve the processing time of pupillometry videos, although there is great advantage for CUDA processing.*
Keywords: *Pupillometry, CUDA, OpenMP, GPU.*

Introdução

A pupila tem como principal função regular a quantidade de luz que atinge a retina. Assim, constrições e dilatações ocorrem primariamente por conta do reflexo à luz e do reflexo de acomodação [1]. Além de suas funções primárias, a dinâmica pupilar é alterada por diversas condições como, por exemplo: estímulos emocionais, processamento cognitivo, aprendizado, comunicação não verbal, sendo também alterada em decorrência de diferentes patologias [1]. Por este motivo, o estudo desta dinâmica é de grande importância para o diagnóstico de diversas doenças que podem causar impactos sociais e alterações cognitivas, ocupacionais e físicas. Estudos recentes têm utilizado séries de pupilometria como exame para a avaliação da fadiga, método de detecção de patologias no sistema nervoso (*Alzheimer*), sistema endócrino (*Diabetes Mellitus*), na detecção de doenças oculares (*Glaucoma*), bem como na identificação de transtornos nos ritmos circadianos (sonolência diurna excessiva) [5].

A pupilometria é uma técnica que permite a medição do diâmetro da pupila possibilitando o estudo de seu comportamento. Exames de pupilometria são realizados por um instrumento - o pupilômetro - que registra em vídeo a variação do diâmetro da pupila ao longo do tempo. Posteriormente, os vídeos gravados são processados com o intuito de se extrair a série temporal de diâmetro pupilar (STDP), o pupilograma. Tendo em vista que a identificação da pupila e seu posterior processamento podem ser computacionalmente custosos, esforços têm sido investidos no aumento do desempenho da execução destas tarefas. Uma das estratégias seria paralelizar o processamento de vídeo.

Neste contexto, duas soluções de computação paralela vêm se mostrando promissoras. A primeira é a biblioteca OpenMP (*Open Multi-Processing*) que tira proveito dos diversos núcleos (*multi-core*) disponíveis em um processador para distribuir a carga de trabalho. A segunda é a plataforma CUDA (*Compute Unified Device Architecture*) que utiliza placas gráficas (GPU - *Graphic Processing Unit*) de uso geral para promover o paralelismo entre processos.

Mesmo tirando proveito das características citadas, o

paralelismo proporcionado por ambas arquiteturas é limitado em diferentes aspectos, especialmente os referentes ao acesso a recursos compartilhados, o qual é realizado de forma sequencial, e a troca de informação entre processos. Assim, os ganhos obtidos dependem do *hardware* usado e da porção de código paralelizado.

Em geral a performance de algoritmos paralelizados com CUDA é superior aos paralelizados com OpenMP [4]. Isto ocorre especialmente devido à quantidade de núcleos que é maior em placas gráficas e pela forma com que a informação é distribuída entre estes núcleos.

Em situações em que o número de *threads* utilizadas pelo OpenMP é consideravelmente maior que o número de núcleos, o tempo de execução paralelo tende a ser superior ao tempo sequencial, devido a fatores tais como troca de contexto e comunicação entre *threads*. O mesmo pode ocorrer na plataforma CUDA, quando a quantidade da informação a ser processada é pequena, pois fatores tais como transferência de dados para a memória da GPU podem fazer com que os tempos de execução sejam próximos ao sequencial [4].

Assim, o objetivo deste trabalho é investigar qual dos métodos (sequencial, OpenMP ou CUDA) propicia um processamento mais eficaz dos vídeos de pupilometria.

Métodos

Vídeos – Neste trabalho, foram utilizados vídeos obtidos de 23 registros de voluntários adultos (idade entre 18 e 30 anos) [6]. Estes vídeos foram coletados no Departamento de Psicobiologia da Universidade Federal de São Paulo (UNIFESP), usando o pupilômetro implementado por Souza *et al.* (2013) [6].

Os vídeos foram armazenados com resolução de 320 x 240 *pixels* a uma taxa de 120 quadros por segundo. A taxa de amostragem é o fator que impossibilita atualmente a análise *online* pelos métodos sequenciais de extração. Este experimento foi aprovado pelo comitê de ética local (UNIFESP; 0763/10) e todos os voluntários assinaram um termo de consentimento.

Pré-processamento – Inicialmente, realizou-se uma filtragem Gaussiana na imagem. O efeito desta operação é a suavização dos contornos da imagem e a diminuição da intensidade do ruído. Após a filtragem, realizou-se a binarização da imagem, seguida pela aplicação de operações de dilatação e erosão com o objetivo de tornar as formas na imagem mais uniformes. Todas as etapas de pré-processamento foram realilzadas utilizando-se a biblioteca OpenCV (*Open Source Computer Vision*).

Extração de contornos - Para identificação de bordas, utilizou-se o detector de *Canny*, método que faz uso dos gradientes (em *x* e *y*) para encontrar regiões com altas frequências na imagem.

Em seguida, foi implementado um método baseado em Suzuki (1985) para a extração de características morfológicas dos contornos, o qual realiza a análise da vizinhança dos *pixels*. Este método baseia-se na existência de uma componente conexa, obtida a partir da varredura da vizinhança-8, iniciando-se por um ponto *p* qualquer, até que se encontre uma região fechada na imagem, a qual representa o contorno da pupila.

Estimação do diâmetro da pupila – Uma vez extraídos os contornos pode-se estimar o seu diâmetro. A estimação é feita de forma indireta, por meio da medida do comprimento da aresta de um retângulo que envolve o contorno da pupila, que é obtido considerando-se os pontos mais externos em cada um dos lados do contorno. Este é um método rápido e eficiente que não envolve transformações nos *pixels*, denominado *Minimum Bounding Box* [3]. Esta técnica permite estimativas precisas, mesmo em situações em que a pupila encontra-se parcialmente ocluída.

Aplicando-se as operações descritas nas seções anteriores em cada quadro de um vídeo de entrada, obtém-se a STDP.

Sequencial vs. paralelo – As operações referentes à extração de contorno foram paralelizadas com CUDA e OpenMP. O processamento foi aplicado a vídeos com 72.000 quadros (registros de 600 s).

Métricas para avaliação de desempenho – A performance na extração de características com base nos métodos paralelos e sequencial, foi comparada por meio de métricas baseadas no tempo de execução e do número de processadores envolvidos. O primeiro é fundamental para se comparar a duração do processamento entre métodos. O segundo, incorpora, em um único índice, o ganho de performance, comparando-se o processamento por um método paralelo com o sequencial.

O *Speed-up* teórico representa o ganho de velocidade na execução de um programa e é modelado pela lei de Amdahl [9], que define que o ganho máximo de velocidade atingível por um algoritmo paralelizado, é calculado por:

$$S(n) = \frac{1}{B+\frac{1}{n}(1-B)} \quad (1)$$

em que *B* é a fração de código sequencial executado e *n* é o número de *threads*.

Por outro lado, o *Speed-up* prático é dado em função do tempo de execução do algoritmo sequencial e paralelo implementados, e representa uma estimativa mais realista de ganho na velocidade, sendo dado por:

$$S = \frac{t_s}{t_p} \quad (2)$$

em que t_s e t_p são, respectivamente, os tempos de execução do algoritmo sequencial e do paralelizado.

Os algoritmos sequencial e paralelos implementados neste trabalho foram executados em um computador com um processador *Intel Core I7* (8 núcleos de processamento), 8 *Giga Bytes* de memória *RAM* e placa Gráfica *Nvidia GeForce* 630M com 2 *Giga Bytes* de memória de vídeo.

A comparação de desempenho foi realizada por meio do teste *Kruskal-Wallis* aplicado ao tempo de execução para os 23 registros com os três diferentes métodos. A

comparação do *Speed-up* para os algoritmos paralelos foi realizada pelo teste *Wilcoxon Rank-Sum*.

Resultados

A Figura 1 ilustra as etapas de processamento implementadas neste trabalho. Os tempos de execução para os três métodos avaliados são apresentados na Figura 2. O algoritmo sequencial apresentou os piores resultados, tendo como mediana 533 s. Os algoritmos paralelizados usando-se respectivamente OpenMP e CUDA apresentaram medianas de 196 e 82 s.

O resultado do teste Kruskall-Wallis apresentou diferença estatisticamente significativa ($p = 7\ x10^{-14}$) para o *Speed-up* e o teste *Post-hoc* de Tukey-Kramer indicou haver diferença entre os três métodos.

Também foram calculados os *Speed-up* para os dois algoritmos paralelizados, sendo o resultado sumarizado nos *boxplot* da Figura 3. O algoritmo paralelizado usando-se as bibliotecas OpenMP e CUDA obtiveram, respectivamente, medianas de 2,7 e 6,5. O Teste de Wilcoxon *Rank-Sum* indicou diferença estatisticamente significativa entre os *Speed-ups* ($p = 6x10^{-9}$).

Para o método OpenMP considerando-se o processador utilizado nos testes (8 *threads*) e a fração de código sequencial a ser executado (23,27%), obteve-se um *Speed-up* teórico de 3,04, 11% maior que o *Speed-up* prático encontrado. Por não existir consenso sobre o cálculo de *Speed-up* teórico em GPUs, o mesmo não foi calculado.

Discussão

O resultado de ambos os testes estatísticos indicaram que os métodos são significativamente diferentes entre si, sugerindo que tanto a plataforma CUDA, quanto a biblioteca OpenMP, atendem o objetivo de agilizar o processamento dos vídeos de pupilometria. Os resultados obtidos com CUDA são superiores aos demais uma vez que o número de *threads* que executam o programa em paralelo é maior que o executado com OpenMP e sequencial. Além disso, deve-se ressaltar que a GPU é otimizada para trabalhar de forma paralela, o que favorece os resultados. Também percebe-se que mesmo com os piores resultados, o algoritmo sequencial apresenta tempos de execução superiores à duração da aquisição de 72.000 quadros (600 s) para a maioria dos vídeos processados, ou seja, o processamento *online* não é garantido para a totalidade de vídeos.

Além disso, cabe salientar que neste trabalho o processamento foi realizado sem a apresentação do vídeo, tarefa que resulta em um aumento significativo no tempo de execução - cerca 465%, 682% e 381%, respectivamente, para os algoritmos sequencial, OpenMP e CUDA. Portanto, no computador empregado e dadas as especificações dos vídeos utilizados (resolução de 320 x 240 e taxa de amostragem de 120 Hz), somente se conseguiria uma análise *online* e apresentação simultânea dos vídeos com o CUDA. Vale ressaltar que o emprego de taxas de amostragem elevadas favorece a identificação de piscadas, cuja duração tem sido considerado um importante parâmetro para análise de estado de alerta/sonolência [7] juntamente com a variação do diâmetro pupilar.

Percebe-se que a diferença entre valor de *Speed-up* médio (2,70) encontrado para o algoritmo OpenMP e o *Speed-up* teórico (3,04) pode ser explicada por uma característica da lei de Amdahl, a qual assume que o sincronismo inter-processos, acesso a dados em memória e outros atrasos no processamento são nulos. Neste caso ideal, não haveria código sequencial a ser executado e o *Speed-up* teórico seria igual ao número de *threads* sendo executadas no processador (conforme Equação 4), gerando uma estimativa otimista. Os valores de *Speed-up* médios encontrados para a plataforma CUDA são cerca de 2,5 vezes maiores que os encontrados ao se utilizar a biblioteca OpenMP. Esta diferença pode assumir valores maiores dependendo da resolução dos vídeos [4].

Percebe-se que a metodologia de extração do diâmetro é robusta, uma vez que se baseia apenas no comprimento da aresta do retângulo que compreende o contorno da pupila, e não em transformações nos *pixels* como, por exemplo, na a transformada de Hough para círculos [2]. Isto permite que o diâmetro seja estimado mesmo quando a pupila está parcialmente ocluída, sem que processamento adicional seja necessário.

Entretanto, quando a oclusão ultrapassa a linha média horizontal da pupila, o diâmetro será subestimado devido ao algoritmo *Minimum Bound Box* [3].

Conclusões

Com base nos resultados encontrados, pode-se concluir que é possível promover o processamento *online* das STDP, tanto com OpenMP quanto com CUDA.

Porém, quando o vídeo é simultaneamente apresentado ao usuário final, apenas o algoritmo paralelizado com CUDA consegue realizar a análise *online*, pois esta tarefa adiciona uma carga significativa de processamento. Assim, ambos os métodos de computação paralela se mostraram mais eficientes quando comparados com o método sequencial, o que encoraja seu uso na análise da dinâmica pupilar.

Agradecimentos

À CAPES, FAPEMIG, CNPq, FAPESP e UFMG/Prpq pelo auxílio financeiro.

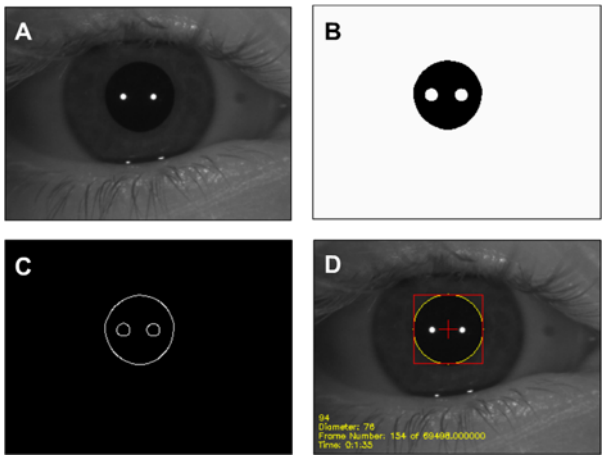

Figura 1: A) Imagem Original; B) Pré-processamento; C) detecção de contorno; D) estimação do Diâmetro.

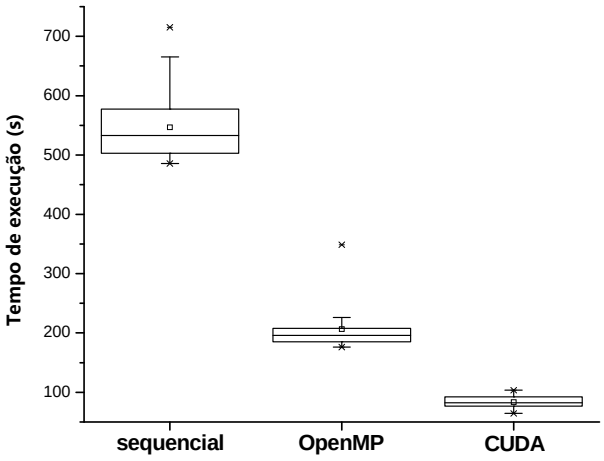

Figura 2: *Boxplot* dos tempos de execução.

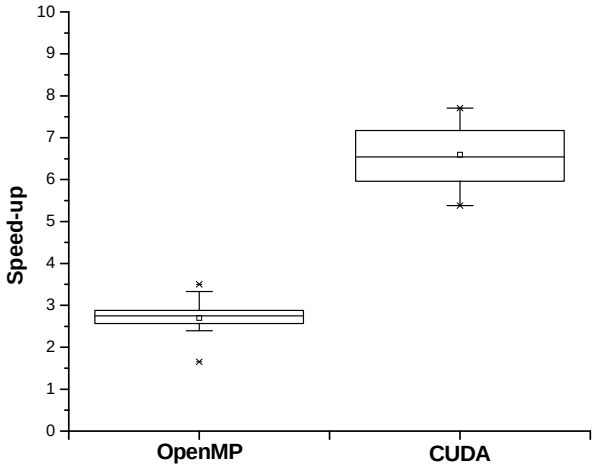

Figura 3: *Boxplot* do *Speed-up*.

Referências

[1] Andreassi J. Human Behavior and Physiological Response. 2ª ed. New Jersey: Psychology Press; 2013.

[2] Gonzales R, Woods RE. Digital Image Processing. 2ª ed. New York: Prentice-Hall; 2011.

[3] Freeman H. Shapira R. Determining the Mini-mum-Area Encasing Rectangle for an Arbitrary Closed Curve: Commun ACM. 1975; 18(7): 409-413.

[4] Körbes A, Vitor GB, Lotufo AR, Ferreira JV. Advances on Watershed Processing on GPU Architectures In: Proc. 10th International Symposium on Mathematical Morphology; Verbania, Itália. 2011. p. 260-271.

[5] Soares VCG, Souza JKS, Ginani GE, Pompéia S, Tierra-Criollo CJ, Melges DB. Identification of drowsiness and alertness conditions by means of Spectral F-Test applied to pupillometric signals. Journal of Physics – Conf. Ser. 2013; 477: 012027.

[6] Souza JKS, Pinto MS, Vieira PG, Baron J, Tierra-Criollo CJ. An open-source, FireWire camera-based, Labview-controlled image acquisition system for automated, dynamic pupillometry and blink detection. Comput Methods Programs Biomed. 2013; 112(3): 607-623.

[7] Souza JKS, Ginani GE, Pompéia S, Baron J, Tierra-Criollo CJ. Equipamento portátil para ensaios simultâneos de teste de vigilância psicomotora e pupilometria. Braz J Biomed Eng. 2012; 29(1): 97-109.

[8] Suzuki S. Topological Structural Analysis of Digitized Binary Images by Border Following. Comput Vision Graph. 1985; 30(1): 32-46.

[9] Tang S, Lee B, He B. Speedup for Multi-Level Parallel Computing. In: Proc. IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum; Shangai, China. 2012. p. 537-546.