REDUÇÃO DE TEMPO DE PROCESSAMENTO DE EXTRATORES DE CARACTERÍSTICAS POR MEIO DE COMPUTAÇÃO PARALELA

William Xavier Maukoski (UEPG) william_maukoski@hotmail.com
Luciano José Senger (UEPG) ljsenger@uepg.br
Lilian Tais de Gouveia (UEPG) ltgouveia@uepg.br

Resumo: A crescente capacidade de armazenamento dos computadores modernos vem aumentando consideravelmente o volume de dados armazenados. Dentre as aplicações da computação, a área de processamento digital de imagens tem um problema inerente a isto, devido ao crescimento da resolução das imagens ao longo dos anos. Nem sempre o aumento do poder de processamento dos computadores modernos consegue ser o suficiente para manter o tempo de resposta baixo ao utilizar tais bases de dados de imagens. Tendo em vista este problema, faz se necessário empregar ou algoritmos com tempo de resposta menores, ou soluções na implementação dos algoritmos que irão ser aplicados a estas bases de dados. Uma destas estratégias é a computação paralela que pode ser utilizada para diminuir o tempo de resposta em alguns casos. Este artigo tem como objetivo investigar a diminuição do tempo de resposta médio em algoritmos extratores de características, conforme aumenta o número de tarefas ao executar estes algoritmos, em um ambiente de alto desempenho.

Palavras chave: Extratores de característica, tempo de resposta, computação paralela.

MINIMIZING COMPUTING TIMES OF FEATURE EXTRACTORS USING PARALLEL COMPUTING

Abstract: The increasing capacity of modern computers have been enlarging the databases. There's an inherent problem in the digital image processing field caused by the images getting larger over the years. The processing capacity of the modern computers isn't necessarily enough to keep fast response for use of these (kind of) databases, what make necessary a fast response algorithms or another solutions for (processing) algorithms for database manipulation. One of the strategies is the parallel computing, which can be used to reduce the response time in some cases. The purpose of this article is investigating the average response time in some characteristic extraction algorithms as number of threads are increased to execute them.

keywords: characteristic extraction algorithm, response time, parallel computing

1. INTRODUÇÃO

A crescente capacidade de armazenamento em computadores atuais, traz a possibilidade de aumentar não só o número de arquivos, como também o tamanho dos mesmos arquivos. A área de PDI (processamento digital de imagens), tem esta característica, tendo não só crescido as bases de dados de imagens, bem como a resolução das imagens.

Por mais que o poder de processamento dos computadores também cresceu muito ao longo da história da computação, nem sempre ele é capaz de conseguir manter o tempo de processamento baixo.

Nestes casos podem ser aplicadas técnicas mais sofisticadas em nível de programação para a diminuição do tempo de resposta. Uma das alternativas é o desenvolvimento de novos algoritmos, os quais seriam menos onerosos computacionalmente, portanto, teriam um tempo de resposta menor. Por mais que esta seja uma solução plausível, tem de se levar em conta que muitos algoritmos foram desenvolvidos a décadas e mesmo após esse espaço de tempo eles ainda são usados por não ter sido desenvolvido até hoje uma alternativa

melhor, isto evidencia a dificuldade inerente de desenvolver algoritmos novos e melhores.

Uma outra solução para diminuir o tempo de resposta é a computação paralela. Que consiste em resolver várias tarefas simultaneamente. Almasi, G.S. e A. Gottlieb (1989). Para a computação paralela ter possibilidade de obter um bom resultado é necessário que a tarefa tenha um bom potencial para ocorrer a paralelização. Este potencial é normalmente trabalhado em volta de algum laço de repetição na estrutura mais dispendiosa de tempo de processamento do algoritmo.

A área de processamento digital de imagens abrange duas grandes áreas, o refino de uma imagem para interpretação humana e a interpretação de imagens para computadores. Suas aplicações variam desde a medicina, interpretação de texto, detecção de objetos em cena até buscadores de internet (MARQUES & VIEIRA, 1999). Por mais simples que um sistema de processamento digital de imagem seja ele tem algumas características fundamentais (BRITTO et al, 2005). Entre estas características uma que é recorrente é a extração de características da imagem, que obtém dados das imagens servindo para identificá-las e assim poder classificá-las corretamente (JAIN et al, 2000). Neste trabalho serão usados algoritmos de extração de características para averiguar se há um ganho na obtenção do tempo de resposta ao utilizar computação paralela em um ambiente que já possui um processador de alto desempenho.

Devido ao fato de imagens serem matrizes, algoritmos extratores de características percorrem elas linha a linha da matriz, o que gera a oportunidade de paralelização da tarefa.

2. Materiais e Métodos:

Para o estudo foi utilizado um computador com um processador i7 3632QM com 2.3 GHz (3.5 GHZ com *turbo Boost*), com 4 núcleos físicos e 4 núcleos virtuais. O computador possuía 8 GB de memória RAM DDR3.

As imagens para o estudo eram 1000 imagens, que variavam em proporção, resolução, cores predominantes e contraste. Essas variações ocorrem porque as imagens são dividias em 10 classes diferentes, com cada classe tendo 100 imagens, sendo essas classes: aborígenes, praias, ruínas gregas, ónibus, dinossauros, elefantes, flores, cavalos, montanhas e comidas. Essas imagens estão disponíveis para download em:

http://www.ppgia.pucpr.br/~alceu/pdi/Imagens/, sendo o arquivo “image.orig.rar”

Foram escolhidos dois extratores de características, para este estudo. O primeiro é o histograma, que é uma maneira de representar a distribuição de frequência de um conjunto de dados previamente definidos dentro de classes (Pearson, 1895). Neste experimento o histograma será a representação da frequência em que uma determinada cor aparece na imagem.

O outro extrator de característica escolhido foi o LBP (*Locally Binary Pattern*), que é um extrator de características voltado para a textura da imagem, nele uma máscara percorre a imagem fazendo a comparação do pixel central da

máscara com os pixels vizinhos, sendo a distância do pixel central para seus vizinhos definida pelo raio da máscara. Caso ele seja maior que seu vizinho, é considerado como 0 o retorno desta comparação, caso não, o retorno será 1, após todos os vizinhos serem comparados tem-se uma sequência de uns e zeros, que pode ser interpretado como um número binário, ao converter ele para decimal, temos o índice de onde essa incidência será representada, no vetor de LBP, similar ao histograma (DC, 1990).

A biblioteca de visão computacional escolhida para manipular as imagens do experimento foi o Opencv, a linguagem de programação escolhida foi C++ e a medição do tempo será feita pelo próprio sistema operacional, ao executar o comando de execução junto da palavra reservada "*time"*.

O experimento será executado 10 vezes para cada uma das variações no número de *tarefa*, será medido seu tempo de resposta para cada uma das execuções após isso será feito o cálculo da média do tempo de resposta, e o desvio padrão do tempo de resposta para cada uma das situações.

Para medição do ganho de tempo, da execução paralela em relação a serial, será usada a medida de S*peedUp*, que pode ser definida como a razão do tempo sequencia pelo tempo da execução paralela, sendo descrito pela equação a seguir:

$$S = \frac{T(1)}{T(N)}$$

Equação 1 - SpeedUP

Onde, S é o *Speed UP*, T(1) é o tempo serial e T(N) é o tempo paralelo.

Além do S*peed UP* será calculado a eficiência, que é outra maneira de avaliar a paralelizarão de um algoritmo. Ela busca mensurar o quanto o número de *threads* está influenciando no ganho, e é descrita pela equação a seguir:

$$E(p) = \frac{S(p)}{p}$$

Equação 2 - Eficiência

Onde E(p) é a eficiencia, S(p) é o *SpeedUP, e p o* número de *threads.*

Após a implementação do algoritmo de extração de características, o eles foram aplicados na base de dados 10 vezes para cada um dos valores de tarefas escolhidas, obtendo um tempo de resposta médio para conseguir mensurar o *SpeedUP* e a eficiência da solução paralela.

Os valores de tarefas escolhidas foram 1, por ser o mesmo tempo de execução de uma versão serial do projeto, 2 tarefas, 4 tarefas,8 tarefas e 16 tarefas. Os números serão em potencia de dois para adequar-se a estruturas de árvores da execução dos tarefas.

3. Resultados e Discussão:

A seguir encontram-se listados os tempos de execução em segundos para cada uma das dez execuções, a média do tempo das execuções, o desvio padrão, o *SpeedUP* e a eficiência da utilização do número de tarefas especificados na coluna.

	1 Processo	2 Processos	4 Processos	8 Processos	16 Processos
1° Execução	49,912	27,76	15,829	13,312	14,575
2° Execução	49,573	27,567	15,938	13,364	14,57
3° Execução	49,633	27,513	15,958	13,405	14,568
4° Execução	49,213	27,945	16,215	15,726	14,808
5° Execução	49,925	27,489	18,61	13,083	14,929
6° Execução	49,454	28,07	17,164	16,083	17,237
7° Execução	50,05	27,488	18,641	15,799	16,124
8° Execução	49,944	27,447	16,13	15,86	14,81
9° Execução	50,405	27,816	18,79	15,945	16,525
10° Execução	50,512	27,978	18,355	16,027	17,681
Tempo médio	49,8621	27,7073	17,163	14,8604	15,5827
Desvio padrão do tempo	0,406818674	0,234961959	1,29289881	1,35710445	1,202575574
Speed Up		1,8	2,9	3,35	3,2
Eficiência		0,9	0,725	0,418	0,2

Tabela 1 tempo, média, desvio padrão, o SpeedUP e a eficiência (Autor)

Pode ser observado que com o aumento do número de tarefas usadas para executar o algoritmo paralelamente há uma tendência a aumentar o desvio padrão do tempo de execução, essas pequenas inconsistências acontecem por estar precisando dedicar mais esforço para a fazer distribuição das tarefas entre os tarefas.

Como pode ser observado na tabela 1, o tempo de execução teve uma pequena variação nas execuções que possuíam o mesmo número de tarefas, tendo um desvio padrão baixo, com no máximo três segundo de variação entre eles (para a execução com 8 tarefas). Essa variação maior pode ser decorrencia de algum processo em segundo plano que pode ter consumido mais recursos computacionais.

Usar 8 tarefas se mostrou a escolha que diminuiu mais o tempo de resposta, ficando claro pelo seu *SpeedUP*, isto ocorre por ser um processador com um total de 8 núcleos, somando os núcleos físicos e virtuais do processador, esse fato faz com que haja um processo para cada núcleo do processador, caso o numero de tarefas aumente em relação ao numero de núcleos, os núcleos vão precisar escalonar entre os tarefas o que irá trazer perda de desempenho, isso fica claro ao vermos os tempos de execução e o *SpeedUp* utilizando 16 tarefas. Destaca-se ainda que apesar de diminuir o tempo de resposta ao utilizar o mesmo numero de tarefas que o numero de nucleos do processador, esta nem sempre é a solução com maior grau de eficiência.

A seguir temos um gráfico apresentando o ganho no *SpeedUp* em relação ao numero de tarefas utilizadas.

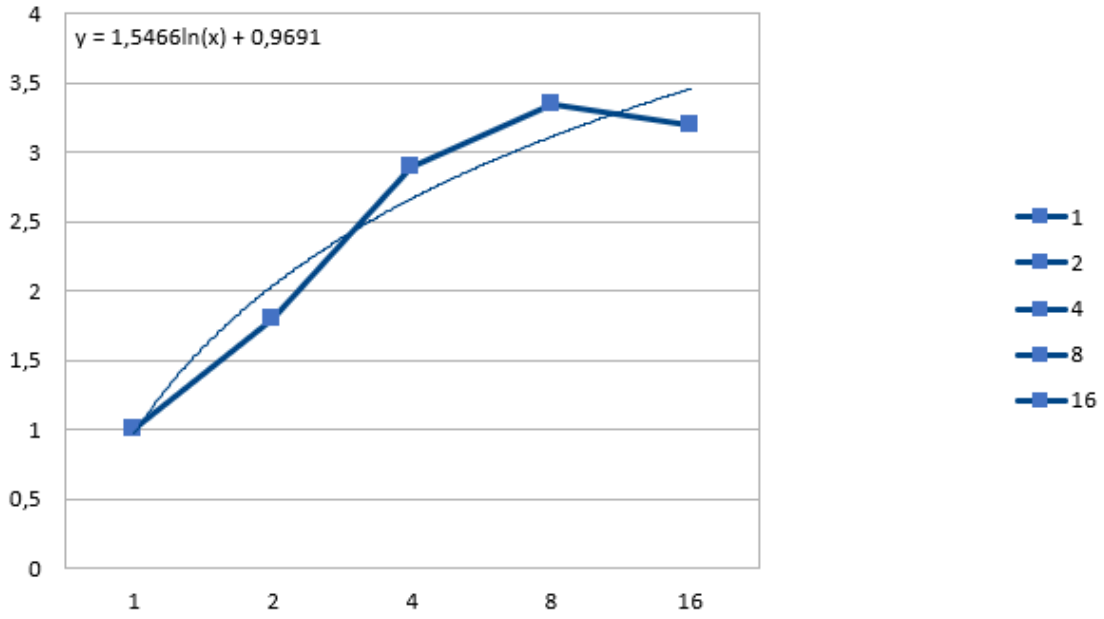

Figura 1 SpeedUP/Processos (Autor)

Do ponto de vista teórico o *SpeedUP* deveria ter um comportamento assintótico tendendo a um valor que ele nunca atingira devido a minimizar o máximo o tempo possível de paralelização, e tender ao tempo de execução impossível de paralelizar. Porém, não é o que ocorre na prática, pois ao aumentar o número de tarefas alem do número de núcleos disponíveis, é necessário que ocorra um escalonamento de tarefas, onde se perde tempo ao faze-lo, conforme cresce o número de tarefas, cresce o tempo necessário para escalonar os tarefas o que diminui o desempenho da solução.

O processador utilizado no experimento é um processador de alto desempenho (i7 3236 QM), o que torna muito claro o ganho de desempenho ao utilizar uma solução paralela com o intuito de reduzir o tempo de resposta na execução dos extratores de característica frente a apenas delegar a tarefa de diminuir o tempo de resposta para o poder de processamento.

4. Conclusão

A utilização de tarefas em número igual ao número de núcleos disponíveis foi a alternativa que apresentou o maior ganho, diminuindo o tempo de resposta dos algoritmos de extração de características. Aumentar acima desse o número de tarefas para a execução do algoritmo trouxe perda de desempenho.

Mais importante do que a capacidade de processamento bruta do processador foi o número de núcleos para escalonar a carga de processo, esse é um ponto que deve ser dado a devida importância na hora de escolher o material para o experimento.

Referencias

Almasi, G.S. e A. Gottlieb (1989). **Highly Parallel Computing**. Benjamin-Cummings, Redwood City, CA.

Corso D. A.; Almeida R. H. P. Britto, JR. A. (2005). Extração de Características Baseadas em Forma para o Reconhecimento de Padrões em um Sistema de

Visão Computacional Disponível em: <http://www.ppgia.pucpr.br/~alceu/pdi/Momentos%20HU/Artigo%20Diego%20e%20Rubens%20-%20Hu.pdf>. Acesso em: 02 Nov. 2017.

DC. He and L. Wang (1990), **Texture Unit, Texture Spectrum, And Texture Analysis**, Geoscience and Remote Sensing, IEEE Transactions on, vol. 28, pp. 509 - 512.

JAIN, A.; DUIN, R.; MAO, J., "**Statistical pattern recognition: A review**", IEEE Transactions on Pattern Analysis and Machine Intelligence, 22, (2000), 4.37.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **Processamento Digital de Imagens**, Rio de Janeiro: Brasport, 1999. ISBN 8574520098

Pearson, K. (1895). "**Contributions to the Mathematical Theory of Evolution. II. Skew Variation in Homogeneous Material**". Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 186: 343–414.