Implementação Paralela de um Algoritmo de Detecção de Objetos em Movimento

César Leonardo B. Silveira, Júlio Cezar Silveira
Jacques Jr, Cláudio R. Jung, Gerson G. H. Cavalheiro

Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Universidade do Vale do Rio dos Sinos
Av. Unisinos 950, 93022-000 São Leopoldo, RS
{cesarls, julioj}@turing.unisinos.br, {crjung, gersonc}@unisinos.br

Introdução

A detecção de objetos em movimento a partir de uma seqüência de imagens é um importante problema no campo da visão computacional. Uma vez identificado um objeto em movimento, é possível estimar como este movimento irá afetar o ambiente que está sendo observado. Diferentes áreas de aplicação da visão computacional, tais como segurança, monitoramento de tráfego, análise do desempenho de atletas e observação de indivíduos em locais públicos, aplicam técnicas de detecção de movimento. No entanto, a precisão nos resultados depende da complexidade do algoritmo aplicado. Quanto mais complexos estes algoritmos forem, maior a demanda de processamento requerida.

A subtração de imagens de fundo (*background*) serve como base para diversas técnicas. No processo de subtração de *background*, o primeiro passo consiste em gerar um modelo de imagem contendo informações que representam o fundo da seqüência analisada. Em seguida, cada quadro da seqüência é comparado com este modelo. O resultado dessa comparação permite identificar os objetos em movimento. Finalmente, a qualidade das imagens geradas é melhorada através da remoção de sombra.

Neste trabalho é proposta a implementação paralela utilizando *multithreading* para o algoritmo W4 [W4 00], cuja implementação seqüencial em Matlab encontra-se em [JUL 05]. Como o algoritmo original possui alto grau de concorrência, um estudo preliminar analisa os diferentes custos computacionais de cada passo envolvido no processo de detecção de objetos em movimento. A concorrência foi explorada quando os custos de execução justificavam os sobre-custos de geração e gerência de paralelismo.

O algoritmo encontrado em [JUL 05] e os custos computacionais de cada etapa são discutidos a seguir. Na seqüência, é apresentada a implementação paralela deste algoritmo, bem como resultados de desempenho.

Algoritmo W4

O primeiro passo do algoritmo é denominado *treinamento*. Este passo é responsável por gerar um modelo de imagem de fundo com o qual cada quadro da seqüência será comparado. Para a realização deste procedimento, é utilizada uma seqüência de amostra V com N quadros da seqüência original. Este procedimento está divido em duas partes. No primeiro, considere V uma lista de N quadros consecutivos representados por

matrizes de intensidades de *pixels*. Para cada *pixel* (i,j) é calculado o desvio padrão σ(i,j) e a mediana λ(i,j) da intensidade do *pixel* (i,j) para todos os N quadros de V. Obtidas as matrizes σ e λ, os *pixels* de cada quadro q que passarem pelo teste

$$|V^q(i,j) - \lambda(i,j)| \leq 2\sigma(i,j) \tag{1}$$

serão selecionados para o segundo estágio do procedimento.

No segundo estágio do procedimento, o modelo de *background* é gerado. Este é composto pelas matrizes m, n e d. As duas primeiras, m e n, contém os valores mínimos e máximos das intensidades de cada *pixel* (i,j). A terceira, d, contém as máximas diferenças de intensidades de cada *pixel* (i,j) entre dois quadros consecutivos. Note que os valores são obtidos a partir dos valores que passaram pelo teste (1). Assim, o modelo de *background* B pode ser representado por $B = \left[\begin{array}{ccc} m(i,j) & n(i,j) & d(i,j) \end{array} \right]$.

Obtido o modelo de *background*, o próximo passo é a subtração do *background* de cada quadro da seqüência. Considere B o modelo de *background* e V uma nova lista de quadros, contendo todos os quadros da seqüência. Para cada quadro q em V, é possível determinar quais *pixels* pertencem ao *background* utilizando o seguinte teste

$$V^q(i,j) > (m(i,j) - k\mu) \;\; \&\& \;\; V^q(i,j) < (n(i,j) + k\mu) \tag{2}$$

onde k é uma constante cujo valor é igual a 2 e μ é o valor mediano de d.

A remoção de sombra é o último passo do processo. Esta é realizada tomando-se, para cada *pixel* V^q(i,j), uma vizinhança de raio r *pixels*. Feito isso, são calculadas as razões das intensidades dos *pixels* da vizinhança que não pertencem ao *background* pelas suas respectivas medianas. Após, é calculado o desvio padrão std destas razões. Também é calculada a razão R do *pixel* (i,j) pela sua respectiva mediana λ(i,j). Se

$$(0.5 < R < 1) \;\; \&\& \;\; (std < 0.05) \tag{3}$$

então o *pixel* V^q(i,j) é considerado sombra.

De todos estes passos, dois foram identificados como sendo mais custosos computacionalmente. O primeiro é a geração do modelo de *background*, devido ao fato de que neste passo são realizadas diversas operações de ponto flutuante, como cálculos de desvio padrão. Também são realizadas operações de ordenação sobre uma série de vetores. O segundo é a remoção de sombra, pois neste passo são realizadas uma série operações sobre a matriz de pixels das imagens e é criado um grande número de outras matrizes menores obtidas a partir da matriz principal. A remoção de *background* não requer grande poder computacional, visto que se constitui de comparações e operações lógicas simples.

Implementação

A aplicação paralela para o algoritmo W4 descrito na seção anterior foi implementada em linguagem C. Foram utilizadas as bibliotecas ImageMagick para operações de carregar e salvar os quadros da seqüência de vídeo e POSIX *threads* para o uso de *multithreading*. A plataforma explorada se baseia numa arquitetura Intel Xeon (dual Xeon c/ tecnologia HyperThreading, 2.4Ghz, 2GB RAM) com o sistema operacional Linux (Gentoo Linux, *kernel* 2.6.10, *gcc* 3.3.6).

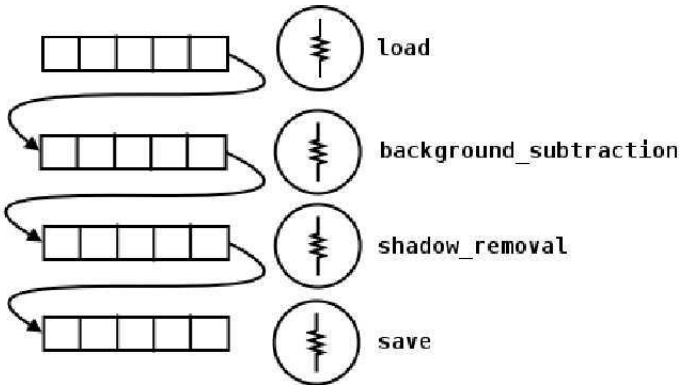

Figura 1: Modelo de execução

Ao ser iniciada a execução, um conjunto de *threads* realiza a geração do modelo de *background*. Cada *thread* é responsável por uma determinada área da imagem. Foi constatado que o número ideal de *threads* para esta etapa é 4. A curva de desempenho pode ser vista na Figura 2a. Este passo é executado apenas no lançamento do programa. Os demais são organizados em uma estrutura semelhante a um *pipeline*. A Figura 1 apresenta um esquema deste modelo de execução.

Para cada passo do algoritmo, existe uma *thread* aguardando trabalhos em uma fila de entrada. Cada *thread* adquire um trabalho, realiza o seu processamento e por fim insere o resultado na fila de entrada do próximo passo. Além das *threads* responsáveis por cada passo do algoritmo, existe uma *thread* responsável por carregar quadros ainda não processados da seqüência e outra responsável por salvar quadros já processados. Para ter controle sobre as filas e quando as *threads* devem pegar trabalhos destas filas, foram utilizadas variáveis de condição. A seguir é feita uma descrição mais detalha do processo.

Após a etapa de obtenção do modelo de *background*, a fila de entrada da *thread* responsável por carregar os quadros da seqüência é preenchida com os nomes de arquivo dos quadros a serem processados, além de informações adicionais como dimensões das imagens e se estão em escala de cinza ou em cores. Esta última informação é necessária, pois caso as imagens não estejam em escala de cinza, as mesmas devem ser convertidas. Todas as informações são carregadas de um arquivo de configuração.

Assim que a *thread* responsável por carregar quadros detecta a presença de novos trabalhos na fila, ela carrega os quadros e divide-os em blocos menores. Feito isso, os novos trabalhos são inseridos na fila de entrada da *thread* responsável pela subtração de *background*. Nesta etapa, caso seja detectado que o trabalho processado não possui uma quantidade significativa de *pixels* não pertencentes ao *background*, o trabalho é inserido diretamente na fila de entrada da *thread* responsável por salvar os quadros processados. Caso contrário, o trabalho é inserido na fila de entrada da *thread* responsável pela remoção de sombra. Por fim, os trabalhos que chegam até a remoção de sombra são processados e inseridos na fila da *thread* responsável por salvar os quadros processados. Um quadro poderá ser salvo quando não existirem mais trabalhos referentes a ele na fila, ou seja, quando todos os seus blocos tiverem sido processados. Quando isto ocorre, o novo quadro é salvo e a memória ocupada pela matriz de *pixels* é liberada. Como mencionado, existe uma *thread* responsável por cada um dos estágios descritos. Foi constatado que

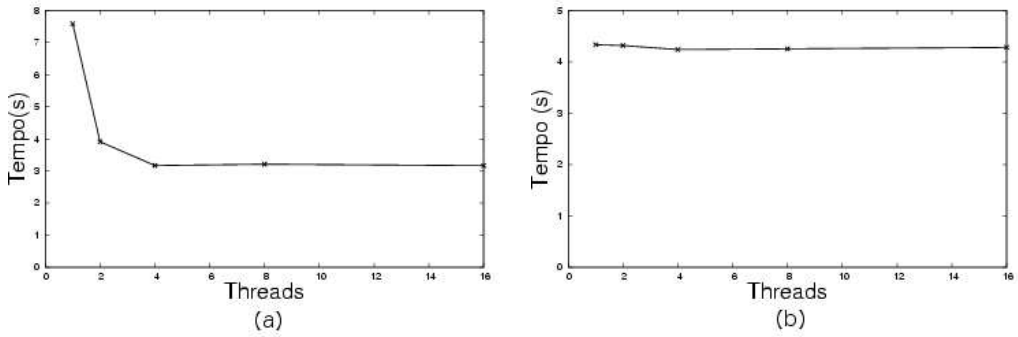

Figura 2: Impacto do número de *threads* no desempenho da etapa de treinamento (a) e no processo de detecção de objetos em movimento (b)

aumentando o número de *threads* responsáveis pela remoção de sombra (etapa identificada como uma das que requerem maior poder computacional), não ocorre um aumento significativo no desempenho do processo. Isto pode ser visto na Figura 2b, que apresenta a curva de desempenho do processo de detecção de objetos em movimento.

É importante notar que os tempos de entrada e saída de dados estão incluídos nos resultados, visto que todas as etapas do processo ocorrem de forma simultânea, incluindo a carga e o salvamento dos quadros. Também é importante notar que, caso as imagens estejam em cores, o processo de convertê-las para escala de cinza requer um certo tempo de processamento. As imagens utilizadas para os testes tinham dimensões de 320x240 *pixels*, estavam em cores e no formato BMP.

Conclusão

Neste artigo foi apresentada a implementação paralela de um algoritmo de detecção de objetos em movimento. Foram apresentados o seu modelo de execução em forma de pipeline e também resultados de desempenho. Os resultados obtidos foram bastante satisfatórios, visto que o tempo gasto durante a detecção dos objetos foi o mesmo da seqüência processada. Logo, o algoritmo foi capaz de processar a seqüência em tempo real.

Referências

[JUL 05] JACQUES JR, Julio Cezar Silveira; JUNG, Cláudio Rosito; MUSSE, Soraia Raupp. Background Subtraction and Shadow Detection in Grayscale Video Sequences. In: SIBGRAPI, 2005, Natal, RN. Proceedings of SIBGRAPI 2005 (XVIII Brazilian Symposium on Computer Graphics and Image Processing). Los Alamitos, USA: IEEE Computer Press, 2005. pp 189–196.

[W4 00] HARITAOGLU, Ismail; HARWOOD, Davis; DAVID, Larry. W4: Real-Time Surveillance of People and Their Activities. In: IEEE Transactions on Pattern Analisis and Machine Intelligence. Washington, DC, USA: IEEE Computer Society, 2000. 22(8): pp 809–830.