Detecção de paralelismo para filtros convolucionais

Bruno Lopes Vieira, Eliana Silva de Almeida,
Alejandro César Frery

Centro de Pesquisa em Matemática Computacional
Universidade Federal de Alagoas

blv@tci.ufal.br, eliana.almeida@pesquisador.cnpq.br,
acfrery@pesquisador.cnpq.br

Resumo

Este trabalho propõe um algoritmo para converter um filtro de imagem convolucional em seu correspondente paralelo. Ele consiste em mapear o código de um filtro de imagens para um grafo de dependências (uma linguagem intermediária), onde se torna possível observar claramente as instruções que podem ser paralelizadas. A partir desse grafo, torna-se possível transcrevê-lo a uma linguagem de programação. Como estudo de caso, utilizou-se um filtro gaussiano codificado na linguagem `R`*, obtendo-se uma redução considerável na quantidade de operações seqüenciais (de* 28 *a* 9 *por pixel).*

Introdução

Filtros de imagens são algoritmos capazes de obter novas imagens a partir de transformações lógicas e/ou matemáticas dos dados de entrada. Podem se apresentar como um conjunto de instruções, tanto na linguagem de circuitos eletrônicos como em um programa de computador.
Além do resultado almejado, um dos pontos críticos na escolha de um filtro é o seu desempenho computacional. Uma abordagem à otimização de algoritmos, é o uso de plataformas paralelas ou distribuídas; os *clusters* de computadores de baixo custo são, em grande parte, responsáveis pela popularização desta abordagem antes limitada a computadores de grande porte dedicados.
Dado o código de um filtro de imagem como entrada, este código é mapeado em uma linguagem intermediária denominada “grafos de dependências” (ver figura 1) Seu uso foi proposto por Ferrante & Ottenstein (1987).

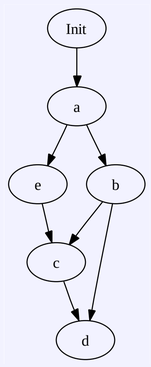

Figura 1: *Exemplo de um grafo de dependências*

A partir desta representação intermediária, é possível produzir o correspondente paralelo do filtro original. Convém salientar que essa metodologia é aprovada pelo IEEE (2006).

Filtros de imagens convolucionais

Uma imagem monoespectral é uma função $f\colon S \to \mathbb{R}$, onde $S = \{0,\ldots,m-1\}\times\{0,\ldots,n-1\}$ é o suporte. O par $(s, f(s))$, com $s \in S$, chama-se pixel, e pode ser conveniente deixar em evidência as componentes da coordenada, isto é, usar (i, j) no lugar de $s \in S$.
Dada a matriz de convolução de lado ℓ ímpar

$$A = (a_{i,j})_{\frac{\ell-1}{2}\leq i,j\leq\frac{\ell-1}{2}},$$

com $a_{i,j} \in \mathbb{R}$, define-se a imagem $g\colon S \to \mathbb{R}$ resultante de filtrar a imagem f por convolução com a máscara A como sendo

$$g(k,\ell) = \begin{cases} \displaystyle\sum_{-\frac{\ell-1}{2}\leq i,j\leq\frac{\ell+1}{2}} a_{i,j}\, f(k-i,\ell-j) & \text{se } (k,\ell)\in S', \\ f(k,\ell) & \text{caso contrário,} \end{cases}$$

onde $S' = \{\frac{\ell-1}{2},\ldots,m-\frac{\ell+1}{2}\}\times\{\frac{\ell-1}{2},\ldots,n-\frac{\ell+1}{2}\}$. Freqüentemente tem-se que $\sum_{i,j} a_{i,j} = 1$.

Algoritmo de mapeamento

O trabalho de Martins (2000) apresenta um estudo de detecção de paralelismo, utilizando-se `grafos de dependências`, porém com base em `semântica denotacional`. Assim sendo, há um algoritmo definido para o mapeamento `semântica denotacional` → `grafo de dependências`. Porém, cada linguagem possui sua semântica; O algoritmo definido é adequado a linguagem `C`.
O algoritmo aqui proposto adequa-se a qualquer linguagem, consistindo nos seguintes passos:

1. Iniciar com o vértice `Init`;
2. interpretar cada comando como um vértice;
3. interpretar cada dependência de controle como uma seta contínua;
4. ao se deperar com um laço `repeat`, caso a condição de saída seja um contador, convertê-lo em um laço `for`;
5. ao se deparar com um laço de repetição `while`, caso a condição de saída seja um contador, convertê-lo em um laço `for`;
6. ao se deparar com um laço de repetição `for`:
   - caso seja possível determinar o número de repetições, gerar uma seqüência paralela para cada repetição;
   - caso não seja possível determinar o número de repetições, gerar apenas uma seqüência com a marca “$*$”;
   - caso os comandos internos em paralelo possuam dependências de dados, seguem em vértices seqüenciais;
7. chamadas a procedimentos seguem em ramos paralelos;
8. caso haja dependência de dados dentre os procedimentos, seguem em vértices seqüenciais;
9. demais casos seguir seqüencialmente.

Filtro gaussiano

Trata-se de uma máscara quadrada cujos coeficientes são proporcionais a uma densidade gaussiana bivariada de média nula e matriz de covariância . Os parâmetros do filtro são o tamanho da máscara e o espalhamento dessa densidade; para filtros definidos sobre máscaras quadradas de lado K ímpar, quanto maior o espalhamento, medido por $\sigma > 0$, maior será o efeito de borramento na imagem filtrada.
Os coeficientes do filtro gaussiano são dados por

$$a_{i,j} = Z_{K,\sigma}\exp\Big\{-\frac{(i^2+j^2)}{2\sigma^2}\Big\},$$

onde $-(K-1)/2 \leq i,j \leq (K-1)/2$ e $Z_{K,\sigma}$ é a constante de padronização que garante $\sum_{i,j} a_{i,j} = 1$.
Este filtro é apropriado para combater ruído aditivo, mas introduz um certo borramento na imagem de saída. Os coeficientes são todos positivos e simétricos em relação ao centro da máscara.
Ao se transpor um filtro gaussiano a um grafo de dependências obtêm-se o resultado exposto na figura 2.

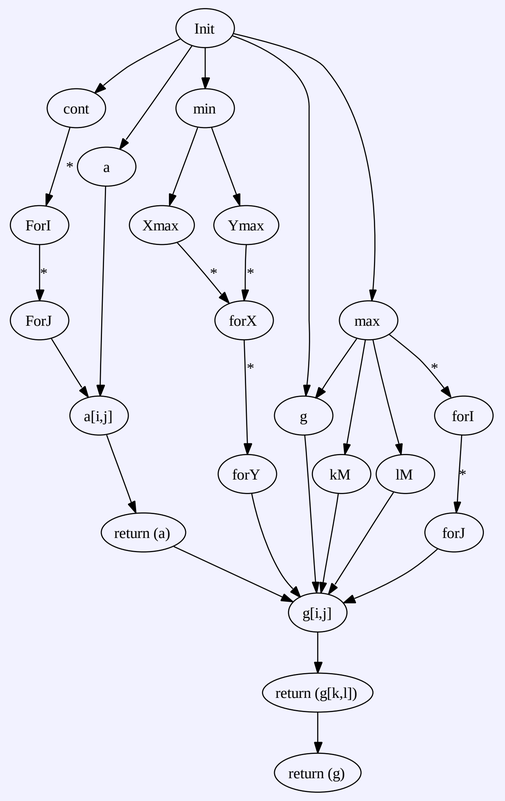

Figura 2: *Filtro gaussiano mapeado*

Resultados

O uso de um grafo de dependências para representar um filtro gaussiano apresenta redução considerável no número de operações seqüenciais, reduzindo-o de 28 a 9 operações por pixel. Em uma imagem com, por exemplo, 1000×1000 pixels, utilizando-se uma máscara de lado $\ell = 3$, a economia em um cluster com cem processadores, o número total de operações seqüênciais se reduz de 3000012 para 30018. Assim sendo, o ganho no tempo de processamento é significativo, e será tanto mais relevante quanto mais interativa for a aplicação, isto é, quando se trata de uma abordagem exploratória.
Com o uso da interface oferecida por Yu (2006), pode-se implementar comunicação do `R` com a interface `MPI` (*Message-Passing Interface*), capaz de gerenciar múltiplos processadores. Assim, o código pode ser facilmente paralelizado na plataforma `R`.

Referências

Ferrante, J. & Ottenstein, K. J. (1987), ‘The program dependence graph and its use in optimization’, *ACM Transactions on Programming Languages and Systems* **9**, 319–349.

IEEE (2006), ‘The world’s leading professional association for the advancement of technology’. URL `http://www.ieee.org`, última consulta em agosto de 2006.

Martins, C. B. (2000), Detecção de paralelismo a partir de semântica denotacional e de grafos de dependência, Dissertação de Mestrado, Departamento de Informática Pontifícia Universidade Católica do Rio de Janeiro.

Yu, H. (2006), *Rmpi: Interface (Wrapper) to MPI (Message-Passing Interface)*. URL `http://www.stats.uwo.ca/faculty/yu/Rmpi`, R package version 0.5-2.