Arquitetura de Sistemas e Computadores

HIAGO OLIVEIRA
Universidade de Évora
31 de Maio de 2014

Objectivo

"Pretende-se com este trabalho desenvolver um conjunto de funções em assembly MIPS para fazer detecção de contornos em imagens a cores. Dada uma imagem RGB, o resultado final deverá ser uma imagem em tons de cinzento, com fundo branco e com traços escuros nos locais onde existem contornos na imagem original."

Introdução

Tem-se como objectivo final do trabalho fazer detecção de contornos de uma imagem 512x512px, convertida *a priori* para formato RGB. Esta imagem será o input usado pelo conjunto de funções para começar a dar forma à imagem final, em escala de cinzas (formato Gray scale) com os contornos devidamente detectados. Devido às limitações do MARS, o tratamento de imagens relativamente grandes demora algum algum tempo a ser feito, neste caso com uma imagem 512 por 512, o programa demora cerca de 1 minuto e meio a ser executado num processador *Intel® Core™ i7-3610QM CPU @ 2.30GHz*. Durante o desenvolvimento do trabalho, foram utilizadas imagens mais pequenas para facilitar o *debugging*, nomeadamente imagens de tamanho 16x16, 64x64, 128x128, sendo que o programa terminava a execução quase imediatamente nestes casos.

Todas as imagens são compostas por pixeis, e em formato RGB, cada pixel composto por 3 bytes que fazem o controlo da intensidade de cada cor, vermelho, verde, e azul. Sendo que cada cor tem reservada para si 1 byte, os valores irão variar entre 0 e 255 (0x00 e 0xFF, respectivamente). Quando carregada em memória, a imagem comporta-se como um array de pixeis, os 3 primeiros bytes correspondem ao primeiro pixel, os 3 seguintes ao segundo, e assim sucessivamente. Sabemos portanto que uma imagem de tamanho 512 por 512, em formato RGB, ocupará em memória 512x512x3 bytes, ou seja, 786432 bytes. Ao converter a imagem para formato Gray Scale, cada pixel da nova imagem irá ser composto apenas por 1 byte, este controlando a intensidade do cinzento, também variando entre valores 0 e 255 (preto e branco, respectivamente). A imagem resultante ocupará então em memória 512x512 bytes, ou seja, 262144 bytes.

Tendo em conta toda esta informação de como são compostas as imagens e como se comportam em memória, podemos proceder ao desenvolvimento do trabalho.

Desenvolvimento

Todo o processo de detecção de contornos é feito com o auxilio das seguintes funções:

1. Read RGB Image
2. RGB to Gray
3. Convolution
4. Contour
5. Write Gray Image

O *flow* do programa segue pela ordem descrita na lista anterior, tendo cada *função* uma função importante em todo este processo.

Read RGB Image - Serve esta função para abrir e guardar em memória a imagem que se pretende converter, guardando o seu conteúdo num *buffer*. Esta função toma como argumentos o caminho em que se encontra o ficheiro ($a0). Após a sua chamada, tem-se então disponível em memória todo o seu conteúdo, podendo se proceder assim aos seguintes passos.

RGB to Gray - Tal como o nome indica, esta função ao ser chamada irá converter a imagem original guardada em memória para uma em escala de cinzas. Tem como argumentos os *buffers* necessários, o da imagem original, e o que será usado para guardar o resultado ($a0 e $a1, respectivamente). Sabendo que cada pixel é composto por 3 bytes, recorre-se a uma algoritmo para fazer a conversão para a escala de cinzas. A formula utilizada é a seguinte:

$$I = 0.30R + 0.59G + 0.11B$$

Neste passo, para se evitar trabalhar com vírgulas flutuantes optou-se por multiplicar por 30, 59 e 11, cada valor respectivo, e porteriormente dividir-se por 100. O pixel resultante na escala de cinzas é obtido pela soma das suas 3 cores após ser-lhes aplicada a fórmula. Poderá agora ser chamada a próxima função.

Convolution - Esta função é a que consome mais tempo e recursos em termos de processamento. Durante a sua execução numa imagem 512x512, serão executadas um conjunto de instruções perto do milhão de vezes. Tem como objectivo a detecção de variações verticais ou horizontais na intensidade da cor, daí a necessidade de ser chamada duas vezes. Toma como argumentos o *buffer* com a imagem em escala de cinzas ($a0), outro com a matriz a ser usada com o operador de Sobel (horizontal, ou vertical)($a1) e outro que será usado para guardar a imagem resultante com as variações na intensidade da cor ($a2). A detecção de variações verticais ou horizontais é feita através da convolução entre a matriz da imagem e a matriz do operador de Sobel. É ainda necessário fazer a divisão por 4 de cada pixel resultante após a convolução. Após a sua execução, tem-se então disponível a imagem necessária para se obter o resultado final, procedendo-se assim à chamada da próxima função.

Contour - O resultado final é obtido através desta função. Esta toma como argumentos 3 *buffers*, um com as variações horizontais após ser aplicado o operador de Sobel ($a0), outro com as variações verticais($a1), e o que é utilizado para guardar a imagem final com todos os contornos detectados ($a2). Este resultado final é obtido através da combinação das duas imagens filtradas, sendo ainda necessário fazer a divisão de cada pixel por 2 e calculando o seu complementar, subtraíndo-o a 255. Tendo agora a imagem final desejada, resta apenas escrevê-la e guarda-la num ficheiro, sendo necessário por isso a chamada da ultima função.

Write Gray Image - Esta função tem como único objectivo guardar a imagem resultante de todo o processo de detecção de contornos. Ela toma como argumentos o caminho desejado para se guardar a imagem final ($a0), o *buffer* com a dita imagem ($a1) e o tamanho deste mesmo ($a2).

Após a chamada de todas estas funções, é criado um ficheiro em formato Gray no local desejado. Resta apenas converter para um formato PNG, JPG TIFF, etc.

Conclusão

Após a realização deste trabalho, ficou-se com um conhecimento importante de como funcionam as imagens, como são guardadas e organizadas em memória e como se aplicam filtros vistos no dia a dia em vários programas.