Trabajo Práctico 1
Procesamiento de imágenes para la detección de bordes en lenguaje ensamblador

Organización del Computador 2

2do. Cuatrimestre 2009

1. Introdución teórica

La detección de bordes en una imagen reduce significativamente la cantidad de información y filtra la información innecesaria, preservando las características fundamentales de la imagen. De allí que los métodos para detectar bordes sean una herramienta muy poderosa para el procesamiento de imágenes. Podemos definir como un borde a los pixels donde la intensidad de la imagen cambia de forma abrupta. Si consideramos una función de intensidad de la imagen, entonces lo que buscamos son saltos en dicha función.

La idea básica detrás de cualquier detector de bordes es el cálculo de un operador local de derivación. Para simplificar, estudiaremos el caso de imágenes en escala de grises. En la figura 1 se puede ver este concepto. En la parte izquierda se puede ver una imagen de una banda clara sobre un fondo oscuro, el perfil a lo largo de una línea horizontal y la primera y segunda derivada de dicho perfil. Se puede observar que el perfil del borde se ha modelado como una discontinuidad suave. Esto tiene en cuenta el hecho de que en las imágenes reales los bordes están ligeramente desenfocados.

Como se puede observar en la figura 1 la primera derivada es positiva para cambio a nivel de gris más claro, negativa en caso contrario y cero en aquellas zonas con nivel de gris uniforme. La segunda derivada presenta valor positivo en la zona oscura de cada borde, valor negativo en la zona clara de cada borde y valor cero en las zonas de valor de gris constante y justo en la posición de los bordes. El valor de la magnitud de la primera derivada nos sirve para detectar la presencia de bordes, mientras que el signo de la segunda derivada nos indica si el pixel pertenece a la zona clara o a la zona oscura. Además, la segunda derivada presenta siempre un cero en el punto medio de la transición. Esto puede ser muy útil para localizar bordes en una imagen.

Aunque lo que llevamos dicho hasta aquí se refiere a perfiles unidimensionales, la extensión a dos dimensiones es inmediata. Simplemente se define el perfil en la dirección perpendicular a la dirección del borde y la interpretación anterior seguirá siendo válida. La primera derivada en cualquier punto de la imagen vendrá dada por la magnitud del gradiente, mientras que la segunda derivada vendrá dada por el operador Laplaciano.

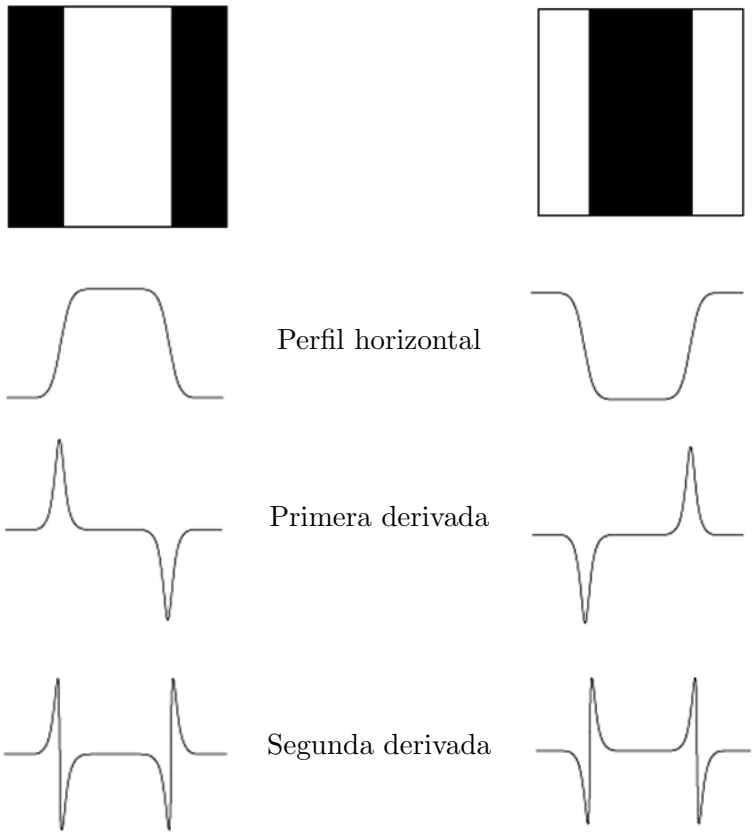

Figura 1: Detección de bordes empleando operadores de derivación. La primera derivada es positiva para cambio a nivel de gris más claro, negativa en caso contrario y cero en zonas con nivel de gris uniforme. La segunda derivada tiene un cero en la posición de cada borde.

El gradiente de una imagen $I(x, y)$ en la posición (x, y) viene dado por el vector:

$$\nabla I = \begin{bmatrix} I_x \\ \\ I_y \end{bmatrix} = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \\ \frac{\partial I}{\partial y} \end{bmatrix}. \tag{1}$$

El vector gradiente siempre apunta en la dirección de la máxima variación de la imagen I en el punto (x, y). Para los métodos de gradiente nos interesa conocer la magnitud de este vector, denominado simplemente como gradiente de la imagen, denotado por $||\nabla I||$ y dado por:

$$||\nabla I|| = \sqrt{I_x^2 + I_y^2}. \tag{2}$$

Esta cantidad representa la variación de la imagen $I(x, y)$ por unidad de distancia en la dirección del vector ∇I. En general, el gradiente de la imagen se suele aproximar mediante la expresión

$$||\nabla I|| \approx |I_x| + |I_y|, \tag{3}$$

que es mucho más simple de implementar computacionalmente.

Para el cálculo del gradiente de la imagen en cada pixel tenemos que obtener las derivadas parciales. Las derivadas se pueden implementar digitalmente de varias formas. Una forma es mediante máscaras de tamaño 3×3 o incluso más grandes. La ventaja de utilizar máscaras grandes es que los errores producidos por efectos del ruido son reducidos mediante promedios locales tomados en los puntos en donde se superpone la máscara. Por otro lado, las máscaras normalmente tienen tamaños impares, de forma que los operadores se encuentran centrados sobre el pixel en el que se calcula el gradiente. El requisito básico de un operador de gradiente es que la suma de los coeficientes de la máscara sea nula, para que la derivada de una zona uniforme de la imagen sea cero.

Los operadores de gradiente común (o gradiente ortogonal) encuentran bordes horizontales y verticales. Estos operadores trabajan mediante convolución, es decir, se aplica una máscara sucesivamente sobre cada píxel de la imagen. Los operadores de Roberts, Prewitt, Sobel y Frei-Chen son operadores dobles o de dos etapas. La detección de bordes se realiza en dos pasos. En el primero se buscan bordes horizontales utilizando la máscara que corresponde a la derivada parcial en x. En el segundo paso se buscan los bordes verticales utilizando la máscara que corresponde a la derivada parcial en y. Finalmente, se suman ambos para obtener el gradiente de la imagen en cada pixel. En la figura 2 se pueden ver los operadores de Roberts, Prewitt, Sobel y Frei-Chen para determinar bordes horizontales (izquierda) y verticales (derecha).

Los operadores de Sobel y de Frei-Chen tienen la ventaja de que proporcionan un suavizado además del efecto de derivación. Ya que la derivación acentúa el ruido, el efecto de suavizado es particularmente interesante, puesto que elimina parte del ruido.

1	0
0	-1

0	1
-1	0

(a)

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

(b)

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

(c)

-1	0	1
$-\sqrt{2}$	0	$\sqrt{2}$
-1	0	1

-1	$-\sqrt{2}$	-1
0	0	0
1	$\sqrt{2}$	1

(d)

Figura 2: Operadores de derivación: (a) de Roberts, (b) de Prewitt, (c) de Sobel y (d) de Frei-Chen.

2. Librería OpenCV

Para simplificar el manejo de los archivos de imágenes, la monocromatización de los archivos y para poder comparar algunos tiempos de ejecución, utilizaremos la librería `openCV`. Para instalar esta librería en las distribuciones de Linux sólo basta con `sudo apt-get install libcv-dev`. Las funciones que utilizaremos de la librería son `cvLoadImage` para cargar la imagen,
`cvCreateImage` para crear una imagen donde poner los resultados intermedios, `cvSobel` para comparar los resultados del procesamiento y `cvSaveImage` para salvar la imagen procesada en un archivo. A continuación describimos brevemente cada función y el formato de imagen que utiliza. Para más información, leer el manual de referencia en
`http://opencv.willowgarage.com/documentation/index.html`. Para poder utilizar la librería debe estar instalada e incluir en el código C lo siguiente:

```
#include <cv.h>
#include <highgui.h>
```

2.1. IplImage

La librería utiliza una estructura para contener las imagenes a procesar. A continuación mostramos esa estructura y comentamos los parámetros importantes a tener en cuenta:

```
typedef struct _IplImage
{
    int  nSize;
    int  ID;
    int  nChannels;
    int  alphaChannel;
    int  depth;
    char colorModel[4];
    char channelSeq[4];
    int  dataOrder;
    int  origin;
    int  align;
    int  width;
    int  height;
    struct _IplROI *roi;
    struct _IplImage *maskROI;
    void  *imageId;
    struct _IplTileInfo *tileInfo;
    int  imageSize;
    char *imageData;
    int  widthStep;
    int  BorderMode[4];
    int  BorderConst[4];
    char *imageDataOrigin;
}
IplImage;
```

De los campos de esta estructura son importantes los siguientes:

- depth: cuantos bits utiliza por pixel (8 bits sin signo, 8 bits con signo, etc).
- origin:0 es arriba a la izquierda (normal), 1 es abajo izquierda (como los bmp).
- width: ancho en pixels.
- height: alto en pixels.
- imageData: puntero adonde comienzan los datos de la imagen.
- widthStep: tamanio de una fila ya alineada (siempre multiplo de 4)

2.2. cvLoadImage

```
IplImage* cvLoadImage(const char* filename, int iscolor=CV_LOAD_IMAGE_COLOR)
```

Para cargar cualquier imagen y monocromatizarla debe escribirse:

```
src = cvLoadImage(filename, CV_LOAD_IMAGE_GRAYSCALE);
```

2.3. cvCreateImage

```
IplImage* cvCreateImage(CvSize size, int depth, int channels)
```

Para crear una imagen para poner el resultado del procesamiento de la imagen src, debe escribirse:

```
dst = cvCreateImage (cvGetSize (src), IPL_DEPTH_8U, 1);
```

2.4. cvSobel

```
void cvSobel(const CvArr* src, CvArr* dst, int xorder, int yorder, int aperture_size=3);
```

Para procesar con Sobel una imagen src con la derivada parcial en x, una máscara de 3×3, y poner el resultado en la imagen dst debe escribirse:

```
cvSobel(src, dst, 1, 0, 3);
```

Para procesar con Sobel una imagen src con la derivada parcial en y, una máscara de 3×3, y poner el resultado en la imagen dst debe escribirse:

```
cvSobel(src, dst, 0, 1, 3);
```

2.5. cvSaveImage

```
int cvSaveImage(const char* filename, const CvArr* image)
```

Para salvar una imagen dst en el archivo "salida.bmp"debe escribirse:

```
cvSaveImage("salida.bmp", dst);
```

3. Medición de performance

Para medir los tiempos de ejecución se utiliza la instrucción rdtsc, que lee el timestamp counter en el registro eax. Para medir desde el lenguaje C, debe ejecutarse la siguiente secuencia de instrucciones:

```
int tscl;
// Tomar estado del TSC antes de iniciar el procesamiento de bordes.
 __asm__ __volatile__ ("rdtsc;mov %%eax,%0" : : "g" (tscl));

....
// Procesamiento de los bordes....
...

// Tomo la medición de tiempo con el TSC y calculo la diferencia. Resultado:
// Cantidad de clocks insumidos por el algoritmo.
 __asm__ __volatile__ ("rdtsc;sub %0,%%eax;mov %%eax,%0" : : "g" (tscl));
return tscl;
```

4. Enunciado

Hacer un programa para el procesamiento simple de imágenes. El programa debe permitir cargar una imagen directamente en escala de grises, aplicar los operadores de derivación de Roberts, de Prewitt y de Sobel para realzar los bordes y guardar la imágen en algún formato que prefieran. Debe estar programado de manera modular, para que sea fácil agregar otros tipos de procesamiento en el futuro.

Se debe escribir la parte de interacción con el usuario y manejo de archivos en lenguaje C, utilizando la librería OpenCV. Las funciones de procesamiento de imágenes deben estar escritas en lenguaje ensamblador. El programa debe recibir en la línea de comandos el nombre del archivo de entrada y la operación:

- r1 = realzar bordes con el operador de Roberts.
- r2 = realzar bordes con el operador de Prewitt.
- r3 = realzar bordes con el operador de Sobel, derivación sólo en x.
- r4 = realzar bordes con el operador de Sobel, derivación sólo en y.
- r5 = realzar bordes con el operador de Sobel, derivación en x y en y.

El prototipo de la función de realzar bordes escrita en lenguaje ensamblador debe ser:

```
void asmSobel(const char* src, char* dst, int ancho, int alto, int xorder, int yorder);
```

También se debe comparar los tiempos de ejecución entre el operador de Sobel implementado por el grupo y el operador de Sobel implementado en la librería.

Informe

Debe reflejar el trabajo hecho para obtener el resultado, las decisiones tomadas (con el estudio de sus alternativas), las estructuras de datos usadas (con gráficos y/o dibujos si ayudan a clarificar), las pruebas que hayan hecho para tomar decisiones o al final para buscar errores en el producto final, etcétera. También se debe incluir los datos de la comparación de tiempos, realizada de forma estadística (el menor tiempo de n corridas, el promedio, etc.). Debe contar como mínimo con los siguientes capítulos: introducción, desarrollo, discusión y conclusiones. Estar estructurado top-down o sea leyendo la introducción se debe saber qué se hizo y cuáles son las partes más importantes. Después de leer los primeros capítulos se debe saber cada cosa que se hizo y como se hicieron las más importantes. En el resto de los capítulos se debe poder leer el detalle de todo lo hecho.

Además, el informe debe incluir:

- Carátula con número del grupo y los nombres de los integrantes con número de libreta y email
- Manual del usuario
- Algunas imágenes con las que hayan probado y su correspondiente procesamiento.
- Instrucciones para el corrector, por ejemplo como ensamblar los archivos fuente para obtener el ejecutable (Makefile).
- Lista de todos los archivos entregados.

El informe se evalúa de manera independiente del código. Puede reprobarse el informe y en tal caso deberá ser reentregado para aprobar la materia.

Entrega

La fecha de entrega de este trabajo es martes 6 de octubre, en el horario de clase (de 17 a 22 hs). La entrega se realizará en un CD que debe incluir, los ejecutables, todos los archivos fuentes necesarios para crearlos, imágenes de prueba con su correspondiente procesamiento y el informe en `pdf`. Si desarrollaron prototipos en lenguaje C para resolver el trabajo primero en alto nivel, deben entregarlos e incluir los resultados obtenidos en el informe. Para ordenar la entrega se deben crear las siguientes carpetas en el CD: `src`, `exe`, `enunciado`, `informe`, `resultados`.