High Performance Computing
Departamento de Ingeniería en Informática
LAB1 : SIMD- SSE

1 Objetivo

El objetivo de este laboratorio es conocer, usar y apreciar las características de paralelismo a nivel de instrucción de un procesador moderno. Muchas veces desconocemos por completo las capacidades de cómputo SIMD que los procesadores poseen, y rara vez nos preocupamos de explotarlas.

En este laboratorio usted implementará la Transformada de Hough, para detección de líneas.

2 La Transformada de Hough

La transformada de Hough es una operación de procesamiento y análisis de imágenes que se utiliza para encontrar instancias de objetos en una imagen binaria. Las dos aplicaciones más comunes de esta tranformada es para detectar/extraer líneas y círculos. Los objetos que se desean encontrar deben ser representados paramétricamente.

Generalmente el proceso completo de extracción de objetos usando la transformada de Hough, incluye etapas de pre y procesamiento. Por ejemplo, si la imagen de entrada es una imagen con niveles de gris, es necesario primero aplicar un filtro de detección de bordes, y luego uno de umbralización (thresholding), y así obtener una imagen binaria (solo con valores cero y 255, por ejemplo). Esta imagen es la entrada a la transformada de Hough.

2.1 Detección de líneas

La transformada de Hough es un mapeo del espacio de la imagen al espacio de parámetros del objeto que se desea extraer.

Suponga que deseamos detectar todas las líneas rectas en la imagen de entrada. La Figura **??** muestra un ejemplo de imagen de entrada. Una representación paramétrica de una línea es

$$y = mx + b$$

Luego, el conjunto de píxeles $\{(x_i, y_i), i = 0, 1, \ldots, \}$ que forman una línea con pendiente m y desplazamiento b en la imagen de entrada, se representa como el punto (m, b) en el espacio de parámetros.

Note que el espacio parámetro debe ser discretizado, es decir, se debe elegir el muestreo en la dimensión m y el muestreo en la dimensión b. Sin embargo, existe un problema fundamental con la dimensión m, pues todas las líneas rectas perpendiculares, tendrán pendiente infinita, lo cual introduce complejidades a la implementación.

Una forma más conveniente de representar paramétricamente una línea es

$$x \cos\theta + y \sin\theta = r$$

donde θ es el ángulo que forma el vector normal de la recta con el eje x, y r es la distancia perpendicular de la recta a la línea que pasa por el origen. La Figura 2 ejemplifica esta representación.

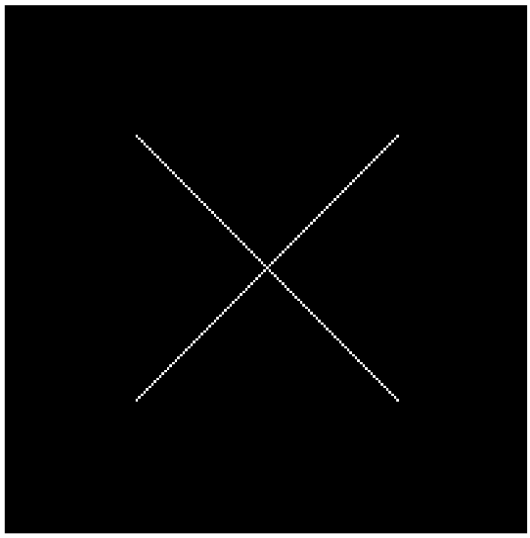

Figure 1: Imagen binaria, de enteros. 255 (blanco), 0 (negro).

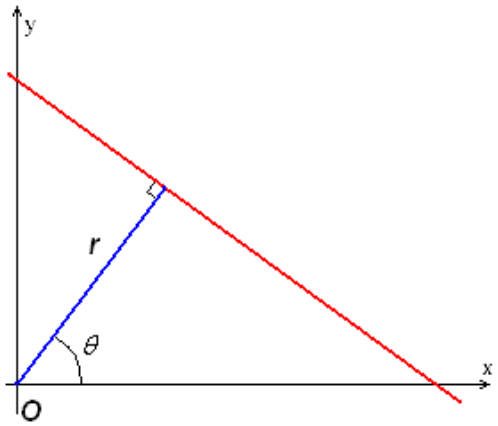

Figure 2: Representación paramétrica (θ, r) de una línea recta.

Luego, para una imagen de $N \times N$ píxeles, se podría elegir la siguiente discretización del espacio paramétrico:

- $\theta_i = i \times \Delta_\theta$, con $\Delta_\theta = \pi/M$ donde M representa el número de ángulos
- $r_j = j \times \Delta_r$, con $\Delta_r = N\sqrt{2}/(2R)$ y R representa el número de desplazamientos..

Un ejemplo concreto de discretización podría ser elegir $M = 180$, y $R = 1000$. Luego, el espacio parámetro, o espacio de Hough sería una matriz de $M \times R$ entradas.

2.2 Detección de líneas

Habiendo elegido el espacio de parámetros y su discretización, el algortimo consiste en recorrer la imagen, pixel a pixel, y para aquellos que pertenecen a alguna línea (aún no sabemos a cuál o a cuáles), incrementar en el espacio parámetro todas las posibles líneas a las cuales podría pertenecer.

El siguiente es un pseudo-algoritmo para este proceso:

```
1    for each pixel (x,y) {
2        if (x,y) is edge {
3            for each theta_i {
4                r_j = x cos(theta_i) + y sin(theta_i)
5                H(theta_i, r_j) = H(theta_i, r_j) + 1
6            }
7        }
8    }
```

Para cada pixel borde de la imagen (pertenece a alguna línea), el algoritmo estima todas las posibles líneas a las cuáles dicho pixel podría pertenecer. Note que todos los píxeles bordes que pertenecen a una línea recta, incrementarán en la misma posición de la matriz H. Luego, al final del procesamiento, H_{ij} contiene el número de píxeles que conforman la línea en (θ_i, r_j).

La Figura 3 muestra un ejemplo de imagen de entrada y su transformada de Hough.

Una etapa posterior consiste en procesar la matriz H mediante una operación de umbralización, buscando todas las posiciones cuyo valor sea mayor que un cierto valor, lo cual detdectaría todas las líneas de largo (en número de píxeles) igual o mayor al umbral.

Esta última etapa se realizará usando acceso directo a memoria, y sin uso SIMD SSE.

3 SIMD SSE

La vectorización de la transformada de Hough puede realizarse de varias, maneras, pero en este lab se usará la técnica de agrupamiento de ángulos.

Este método consiste en agrupar en dos vectors SSE, senos y cosenos, respectivamente, para varios ángulos. Así el loop 3, se debe "desenrrollar" (loop-unrolling), y el cálculo de la línea 4, se realiza mediante operaciones vectoriales.

El incremento de la matriz H se hace directamente accesando memoria sin uso de registros MMX.

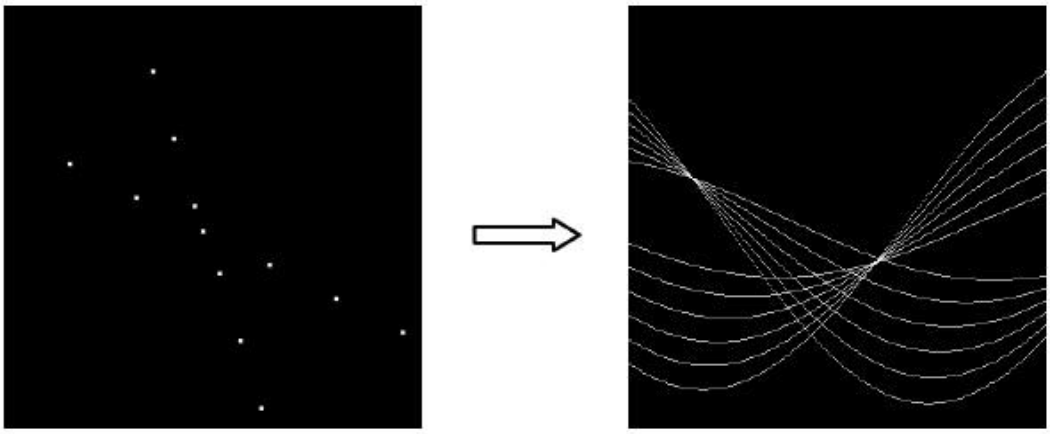

Figure 3: Imagen binaria y su transformada de Hough.

4 El programa

Usted debe construir un programa compuesto de dos partes principales:

1. Cálculo secuencial.
2. Cálculo paralelo-vectorial, con registros MMX.

Para cada parte, debe medir el tiempo de ejecución, usando el llamado al sistema `clock()`:

```
#include <time.h>
clock_t clock(void);
```

El programa debe ejecutarse de la siguiente forma:

```
$ ./hough -i imagen.raw -o hough.raw -N tamaño_imagen -T numero_angulos -R numero_distancias -
```

donde las opciones indican lo siguiente:

- `-i`: imagen de entrada, con enteros (`int`) 0, o 255, en formato binario (raw).
- `-o`: imagen de salida con la transformada de Hough, con enteros (`int`), en formato binario
- `-N`: Tamaño de la imagen (cuadrada)
- `-T`: Número de ángulos
- `-R`: Número de distancias
- `-U`: Umbral para detección de líneas

Además de escribir la matriz H en archivo, el programa debe imprimir por `stdout`:

1. Tiempo de latencia de la parte secuencial

2. Ángulo (valor entre 0 y 180) y distancia para cada línea encontrada

3. Tiempo de latencia para la parte paralela

4. Ángulo (valor entre 0 y 180) y distancia para cada línea encontrada

Para leer y escribir en binario utilice los llamados al sistema `read()` y `write()`. Para visualizar la imagen de entrada y de salida, puede utilizar Matlab. Por ejemplo, suponga `imagen1.raw` es la imagen de entrada. En Matlab, la podemos leer y visualizar de la siguiente manera:

```
f = fopen('imagen1.raw', 'r');   % abre el archivo
I = fread(f, 'int');             % lee todos los enteros
fclose(f);                       % cierra el archivo
I = reshape(I, N, N);            % da forma matricial al vector
I = I';                          % traspone la matriz
imagesc(I); axis('square');colormap(gray); % visualiza
```

5 Entregables

Tarree, comprima y envíe a `fernando.rannou@usach.cl` al menos los siguientes archivos:

1. `Makefile`: archivo para make que compila los programas

2. `hough.c`: archivo con el código. Puede incluir otros archivos fuentes.

Fecha de entrega:
Lunes ?? de septiembre en hora de clases.