Procesadores gráficos y Aplicaciones en Tiempo Real

Box filter de una imagen

Descripción

En esta práctica se abordará el problema de realizar un box filter especifico a una imagen de forma paralela (adaptada de la práctica de blur de *Udacity* del curso *Introduction to Parallel Programming*). En este caso, el filtro de la imagen servirá tanto para mejorar la nitidez de una imagen, cómo suavizarla, o encontrar sus bordes. La idea final es permitir una mejor identificación de los objetos que se encuentren en la imagen.

Para ello se va a realizar un filtrado en el dominio del espacio llevándose a cabo directamente sobre los píxeles de la imagen. En este proceso se relaciona, para todos y cada uno de los puntos de la imagen, un conjunto de píxeles próximos al píxel objetivo sobre los que se aplica un filtro. Para ello se realiza una convolución del filtro (f) sobre la imagen (w), donde cada píxel de la nueva imagen (g) se obtiene mediante el sumatorio de la multiplicación del filtro por los píxeles contiguos:

$$g(x,y) = \sum\sum f(i,j)\, w(i,j)$$

En concreto, imaginemos que tenemos una matriz cuadrada de pesos que representa el filtro. Para cada píxel de la imagen w, superponemos esta matriz cuadrada de pesos de manera que el centro de la matriz se alinee con el píxel a analizar. Para calcular el valor del nuevo píxel, multiplicamos cada par de números que se alinean. En otras palabras, multiplicamos cada peso con el valor del píxel que se ha superpuesto. Por último, sumamos todos los números multiplicados y asignamos ese valor como nuevo píxel. Este proceso se debe realizar para todos los píxeles de la imagen. Esta forma de filtrado es llevada a cabo por herramientas comerciales, por ejemplo Photoshop bajo el nombre “Custom Filter”. La idea de la práctica es su realización de forma paralela.

Para la implementación del box filter se cuenta con un código de apoyo basado en la librería OpenCV. OpenCV es una librería BSD multiplataforma de visión artificial para el tratamiento de imágenes, y se utiliza para abrir, salvar,… y gestionar diferentes tipos de imágenes. El alumno solo debe modificar el fichero *func.cu* para incluir su solución, aunque deberá preparar el proyecto de la forma adecuada para la utilización de OpenCV. Entre otros aspectos se debe modificar en Visual Studio (para la utilización en Linux se facilita un Makefile):

- C/C++-> General -> Additional Include Directories
- Linker -> General -> Additional Library Directories
- Linker -> Input -> Additional Dependencies (en este caso añadir *opencv_core2411d.lib*, *opencv_imgproc2411d.lib* y *opencv_highguid_2411d.lib*. *d* significa libreria para debug. Si se utiliza en modo Realease se debe utilizar la librería normal sin *d*)

En concreto en el código de apoyo, cómo las imágenes en color están formadas por varios canales, se separan los diferentes canales de color de manera que cada color se almacene de

forma separada en lugar de estar intercalados. Esto simplifica el trabajo a realizar, ya que el kernel en CUDA trabajará sólo sobre un único canal de color, aunque habrá que ejecutarlo varias veces para tratar cada canal por separado (el código que recombina los resultados para cada color se encuentra cómo código de apoyo). El objetivo es rellenar el kernel llamado *box_filter* para realizar el filtrado con la matriz de pesos facilitada como *filter* (de *filterWidth* dimensiones cuadradas) sobre el canal de color especificado por InputChannel, escribiendo el resultado en OutputChannel.

He aquí un ejemplo del cálculo que se debe realizar para un único píxel (la matriz de pesos estará alineada para el centro de la caja):

Filtro:

0	-0.25	0
-0.25	2	-0.25
0	-0.25	0

Imagen:

1	2	5	2	0	3
3	2	5	1	6	0
4	3	6	2	1	4
0	4	0	3	4	5
4	5	6	2	3	1

Valor obtenido = 0*2 -0.25*5 + 0*1 – 0.25*3 +2*6 – 0.25*2 + 0*4 -0.25*0 + 0*3 = 9.5

Para su realización de forma paralela, un buen punto de partida consiste en asignar cada thread a un píxel. Entonces, cada thread puede realizar la suma y multiplicación con total independencia del resto de threads, aunque se puede beneficiar de ellos a la hora de conseguir los datos que necesita, por ejemplo mediante la utilización de memoria dinámica.

Objetivos parciales

- Ser capaz de realizar un código paralelo de *box filter*. Para su correcta evaluación, se testeará respecto a la solución de referencia presentada con el filtro por defecto (Laplace 5x5). Si cualquier píxel difiere en más de un cierto valor de umbral, se considerará un error (4.5 puntos).

- Crear y cambiar el filtro para descubrir cómo conseguir nitidez, detección de bordes y suavizados (1 puntos). Como ayuda se puede consultar:

  http://help.arcgis.com/es/arcgisdesktop/10.0/help/index.html#//009t0000004s000000

- Correcta adecuación del número de bloques y de threads por bloque para diferentes tamaños de imagen. No es necesario tener en cuenta la arquitectura concreta de la tarjeta, pero si se debe especificar las limitaciones sobre la arquitectura en concreto sobre la que se ha ejecutado (1 punto).

- Utilización de memoria de constantes para mejorar el rendimiento (1 punto).

- Utilización de memoria compartida y cooperación entre los diferentes threads para mejorar el rendimiento (2.5 puntos).

Nota: las puntuaciones para cada objetivo parcial son las puntuaciones máximas que se pueden obtener si se cumplen esos objetivos. No se debe hacer un programa separado para cada objetivo, sino un único programa genérico que cumpla con todos los objetivos simultáneamente.

Entrega de prácticas

La entrega de prácticas se hará a través del Campus Virtual en las fechas anunciadas. Se debe entregar un único archivo *func.cu* con el código de toda la práctica, debidamente comentado y una memoria de la misma en formato pdf. La memoria debe incluir:

- Índice de contenidos y Autores.
- Descripción del código: incluyendo descripción de las principales funciones implementadas
- Análisis de las mejoras introducidas, especificando tiempos relativos al uso de memoria compartida, memoria de constantes, tamaños de bloque, etc…
- Comentarios personales: incluyendo problemas encontrados, críticas constructiva, propuesta de mejoras y evaluación del tiempo dedicado.
- No incluir código fuente
- NO DESCUIDE LA CALIDAD DE LA MEMORIA DE SU PRÁCTICA. Aprobar la memoria es tan imprescindible para aprobar la práctica, como el correcto funcionamiento de la misma. Si al evaluar la memoria se considera que no alcanza el mínimo admisible, la práctica se considerará SUSPENSA.

Autoría de la práctica

La práctica se debe realizar en grupos de 2 personas, como máximo, pudiendo extenderse hasta 3 personas si se cuenta con algún miembro de la rama no técnica. El hecho de detectar copia en las prácticas expondrá a los alumnos a la posibilidad de una apertura de expediente disciplinario y expulsión. En caso de detectar copia, los alumnos afectados serán suspendidos en la TOTALIDAD de la asignatura. Una práctica será considerada copia en caso de que contenga la totalidad o una parte de la práctica de otro alumno. Se considerará copia en caso de que contengan:

- Archivos con la totalidad o fragmentos de código que no sean del propio autor
- Memorias con la totalidad o fragmentos de frases que no sean de los autores