ESCUELA NACIONAL DE ESTUDIOS SUPERIORES
UNIDAD MORELIA

Computo distribuido

Profesor: Vicente Rodríguez

Proyecto final

Seam Carving: Redimensionamiento de imágenes

David Calderón Ceja
Jorge Antonio Camarena Pliego
Keshava Tonathiu Sánchez Barbosa

06/06/2019

Semestre 02-2019

Introducción

El redimensionamiento de imágenes no debería constar simplemente de restricciones geométricas, sino debería tomar en cuenta el contenido que se tiene presente en la imagen.

Una solución a este problema se puede obtener aplicando un operador llamado “seam carving” a la imagen, el cual opera siendo “consciente” del contenido que esta incluido en la imagen.

Una costura (“seam” traducido al español) es un camino optimo de 8 pixeles conectados en una única imagen, de arriba abajo o de izquierda a derecha, el cual es definido por una función que obtiene la “energía” de la imagen.

Al tallar (“carving” traducido al español) repetidamente la imagen insertando o removiendo costuras podemos cambiar la proporción de una imagen.
La selección y el orden de costuras protegen el contenido de la imagen gracias a la función de energía en la que se apoya.

Explicación del algoritmo

A grandes rasgos el algoritmo que se sigue para realizar el “seam carving” es el siguiente:

1. Asignar el valor de energía a cada píxel de la imagen
2. Encontrar un camino conectado de 8 pixeles (seam) con la menor energía de todas
3. Eliminar todos los pixeles en dicho camino
4. Repetir los pasos 1-3 hasta que se haya eliminado el numero deseado de filas/columnas.

El mapa de energía

El primer paso de nuestro algoritmo consta de calcular un valor de energía para cada píxel individual presente en la imagen, para hacer esto usamos la ecuación más sencilla posible:

$$e_1(\mathbf{I}) = |\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|$$

En esta ecuación ***I*** es nuestra imagen y para cada píxel de la imagen hacemos lo siguiente:

- Se encuentra la derivada parcial del eje x
- Se encuentra la derivada parcial del eje y
- Se suma el valor absoluto de ambos

Esto será el valor de cada píxel, sin embargo, para poder obtener la derivada de una imagen nos apoyamos del “Filtro de Sobel”.

$$p'_u = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{G} \quad \text{and} \quad p'_v = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{G}$$

Detección y eliminación de “seams” con la menor cantidad de energía

Nuestro objetivo ahora que tenemos el mapa de energía de la imagen es encontrar un camino desde la parte superior de la imagen hasta la parte inferior con la menor cantidad de energía. Esta línea debe ser conectada por 8 pixeles en cada píxel que conforma la línea.

Para encontrar esta línea hacemos uso de la siguiente ecuación:

$$\begin{aligned} M(i,j) &= e(i,j) + \\ &\quad \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1)) \end{aligned}$$

Así que creamos un arreglo de 2 dimensiones para almacenar el valor mínimo de energía hasta el último píxel revisado, así tenemos el camino de menor energía requerido para ir desde la parte superior hasta la inferior de la imagen se almacena en la primera fila de la matriz.

De esta manera teniendo los pixeles a eliminar repetimos este método para eliminar el numero deseado de columnas.

Tras operar con la imagen de esta manera, haciendo uso de pillow exportamos la imagen redimensionada.

Paralelizado

Ahora se llevó a cabo paralelización del algoritmo descrito con anterioridad, para esto usaremos una librería de Python llamada “Pycuda” que sirve para computo con GPU, el código se probó sobre una tarjeta de video Nvidia GTX1070 TI y una Nvidia GTX 1080.

Los algoritmos que se paralelizaron para realizar “seam carving” fueron:

1. Convolución.
2. Suma de convoluciones.
3. Búsqueda del seam de menor energía.
4. Retirar seam de menor energía.

Convolución

En el código lo primero que se realiza es leer imagen y descomponerla en sus 3 canales RGB para aplicar su convolución de manera individual, usando el Filtro de “Sobel”. Se aplica primero el filtro P’u en un Kernel construido llamado “convolution” que se lleva como parámetros los canales RGB, anchura y altura de la imagen, una vez en el kernel se asigna un píxel a cada hilo disponible para luego aplicar el filtro P’u, después almacenamos el resultado en nuevo arreglo que es mapa de energía del filtro P’u para cada canal del RGB.

Se repite el proceso anterior pero ahora con el filtro P’v, obteniendo así un mapa de energía con un filtro P’v, el siguiente paso es construir el Filtro de Sobel” que se obtiene sumando el valor absoluto del mapa de energía con el filtro P’u más el valor absoluto del mapa de energía con el filtro P’v de cada canal correspondiente.

Suma de convoluciones

Ya que tenemos el mapa de energía de cada canal RGB falta juntarlo para tener un único mapa de energía, para esto se implementó un kernel llamando “Suma”. Su función es recibir los mapas de los diferentes canales y homogeneizarlo en un solo mapa y para esto se usó la siguiente estrategia, a cada hilo se le asigna un índice el mapa correspondiente a ubicación de un píxel de tal forma que este toma el valor de su índice para sumarlos con el valor de del mismo índice de cada canal, luego se almacena en el índice correspondiente en el mapa de energía homogeneizado.

Encontrar el seam de menor energía

Una vez obtenido el mapa energía se requiere encontrar el seam de menor energía para su extracción. Por esto se implementó la función del kernel llamado “find_min”; su función es: de cada pixel del mapa energía encontrar el pixel de menos energía en sus vecinos de una fila debajo de este para así generar un mapa de índices mínimos como ya se explicó con anterioridad, la función divide y asigna índices a cada hilo del GPU de esta manera cada uno checa el índice menor que le corresponde para obtener el mapa de índices mínimos.

Ya que se generó el mapa de índices mínimos solo hay que saber cuál de todos estos genera el camino con la menor energía, para lograr esto se hizo la función “get_sum_map”, el propósito de esta función es encontrar el índice inicial a seguir para saber qué será cortado de la imagen, para esto la función requiere el mapa de índices mínimos y el mapa de energía.

Cada vez que se llama esta función, se usa el mapa de índices mínimos y genera un nuevo mapa de índices mínimos por reducción y un nuevo camino a seguir para así quedarse únicamente con la primera fila que resumirá todas las energías cumulativas del mapa energía.

Retirar seam de menor energía

Sabiendo cuál es seam de menor energía solo falta extraerlo de la imagen, lo cual se hará usando las funciones implementadas en los kernels llamados “extract_seam_path” y “remove_seam”. A continuación, se explicará cada función.

“extract_seam_path” se usa para que, una vez obtenidos la lista de índices mínimos y el camino a recorrer, se pueda construir un camino de índices mínimos para remover.

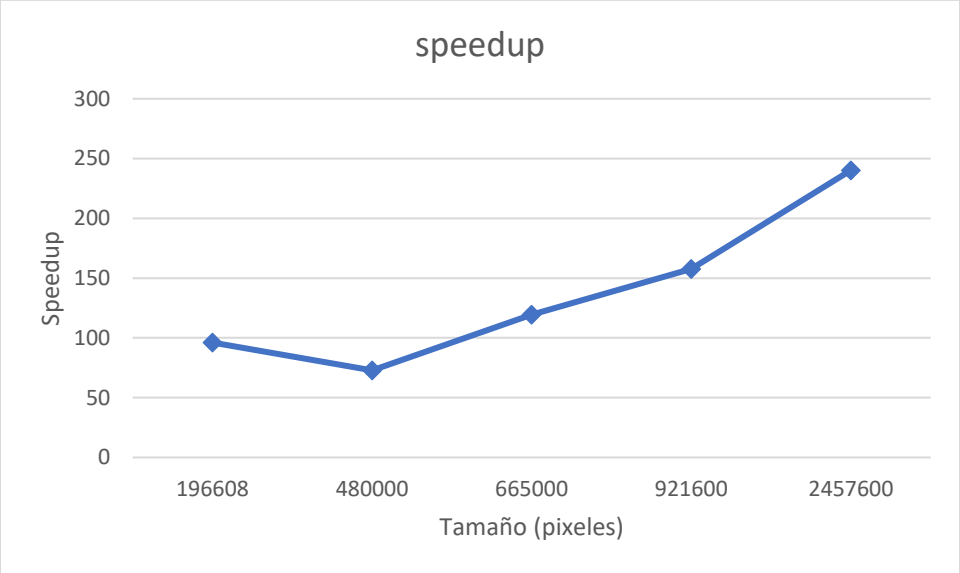
“remove_seam” este kernel se encarga de construir una imagen para cada canal RGB pero con el seam ya removido, el método que se usó para esta función fue que a cada hilo del GPU se le asigna un índice para la imagen, usando los índices

de los canales RGB se reconstruyen en un nuevo canal RGB, se verifica si el pixel de cada hilo esta después del índice a remover, si es así, se recorre el índice del pixel para sustituir el pixel eliminado para cada columna, esto regresa un nuevo canal RGB más pequeño.

De esta manera repetimos la ejecución de los kernels anteriores las veces que sean necesarias para remover el número de seams deseados.

Speedup

¿Qué tanto mejora el “seam carving” al ser paralelizado en pycuda?
Bastante.



La forma de calcular el speedup fue tiempo serial entre tiempo paralelo, podemos apreciar que inclusive en la imagen de menor ganancia (196608 pixeles) obtenemos una mejora de 96 veces el desempeño, y la mayor ganancia de desempeño fue en la imagen más grande (2457600 pixeles) fue de 240 veces más rápido que el serial. A continuación, presentamos una tabla con las imágenes de prueba y el desempeño de ambos algoritmos (grafica anterior basada en la tabla).

Imagen	Tamaño (pixeles)	Tiempo Serial (segundos)	Tiempo paralelo (segundos)	speedup
Birds	196608	96	1	96
Dubai	480000	437	6	72.8333333
Tree	665000	835	7	119.285714
Globo	921600	1419	9	157.666667
Nevado	2457600	3843	16	240.1875

Resultados