Luis Miguel Benitez Ruiz. 994361

Reporte de Lectura

‘Real-Time Object Detection for Multi-Camera on Heterogeneous Parallel Processing Systems’

La detección de objetos en imágenes es una aplicación muy común, pero el rendimiento suele ser bajo, por lo que lograr una detección de precisión aceptable en tiempo real es muy costoso. En el artículo se explica la forma en que se adapto un algoritmo para detección de objetos en video utilizando un esquema de cómputo heterogéneo.

El algoritmo utilizado para la detección no utiliza un clasificador, y consiste en tres pasos:

1) Filtrar los pixeles en base al color. Se establece previamente un rango de colores relevantes, de acuerdo al objeto que se quiere detectar. Se crea una matriz del mismo tamaño que la imagen de entrada. Esta matriz se llena con 1 en las posiciones que correspondan a los pixeles de la imagen de entrada que estén dentro del rango de colores, y 0 en caso contrario.
2) Trabajando sobre la matriz resultado del filtro de color, los pixeles individuales son agrupados basándose en sus vecinos izquierdo y su vecino superior, y cada grupo resultante será considerado un objeto ‘candidato’. Un pixel se asocia a un grupo nuevo si su valor es 1, y sus vecinos a la izquierda y arriba son 0, si alguno de sus vecinos ya pertenece a un grupo, entonces se le intenta asociar primero al mismo grupo que al vecino superior, y después al vecino de la izquierda.
3) Un filtro de segmentos elimina los objetos que no se encuentren en el tamaño esperado, esto también depende del objeto que se está buscando.
4) Cada objeto se analiza y se selecciona el de la forma más cercana al objeto que se está buscando.

De acuerdo al análisis realizado por los autores con base en un programa secuencial, el 46% del tiempo del programa se utiliza en la fase de agrupamiento de objetos, y el 21% en la fase del filtro de colores. El filtro de segmentos y el análisis de objetos candidatos (pasos 3 y 4) solo utilizan 0.06% del tiempo total de ejecución. El resto del tiempo se utiliza en I/O, conversiones de formatos, y otras actividades ‘utilitarias’.

Los autores decidieron dividir la ejecución del algoritmo entre un CPU Intel i7 con 4 núcleos y un GPU NVIDIA Tesla con 448 núcleos. La razón por la que la ejecución es heterogénea en vez de completamente paralela es porque hay segmentos del algoritmo donde se utilizan muchas instrucciones condicionales, por lo que un CPU con múltiples núcleos resulta una mejor forma de paralelizacion que un GPU.

El proceso desarrollado, llamado ‘Scissor Algorithm’ por los autores, es el siguiente:

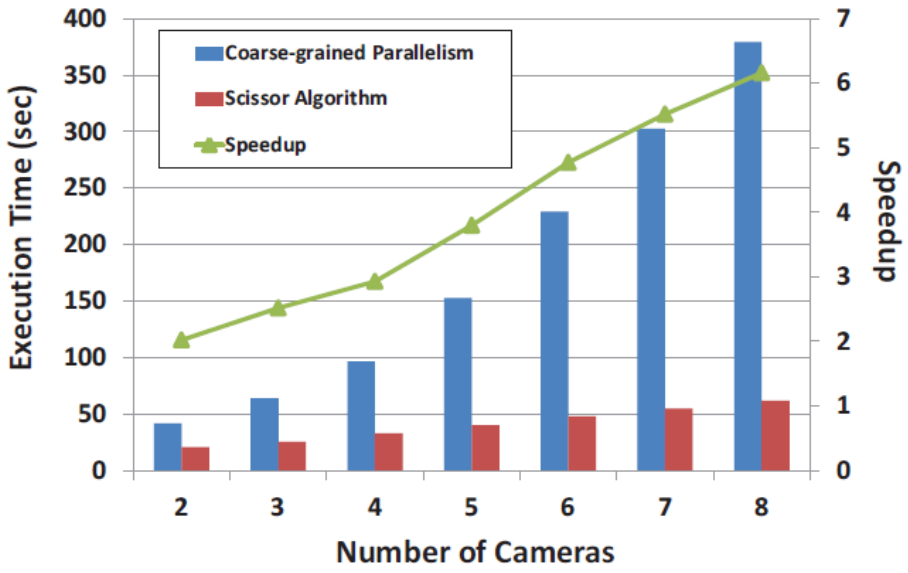
1) Se copian las imágenes de entrada de la memoria principal a la del GPU, donde como primer paso se realiza la conversión de formato (esto no se detalla mas en el articulo)

2) De forma paralela, se revisan los pixeles, se compara si están en el rango de colores relevantes y se escribe 1 o 0 a la matriz CF (en memoria del dispositivo). Esto es el filtro de color.
3) Se sincronizan los threads.
4) De forma paralela se crean dos vectores, ProVecx y ProVecy, que contienen un 1 en los índices correspondientes a los renglones (en ProVecx) y columnas (en ProVecy) que contengan un 1 en CF.
5) Se transfiere CF, ProVecx y ProVecy a la memoria del dispositivo.

Después, en el CPU, se crean los grupos basándose en la información en ProVecx y ProVecy, que ya contienen los índices donde CF es 1.

Los autores experimentaron con paralelismo 'standard', es decir, ejecutando todos los pasos del algoritmo en un CPU multi-hilo, y con el modelo heterogéneo anteriormente descrito. De acuerdo a sus resultados, el modelo heterogéneo tiene un rendimiento casi constante cuando aumenta el número de de cámaras. Para decidir los parámetros de dimensiones de los bloques de hilos se utilizo un método de autotuning, y los autores determinaron que las dimensiones de bloque ideales son de (40,9). El GPU en cuestión tiene 14 SMs con 32 núcleos cada uno.

La grafica muestra los resultados de los experimentos comparando los dos modelos de paralelizacion de este problema:



Referencias

Chi-Sheng Lin, S.-M. T.-T.-A. (2013). Real Time Object Detection for Multi-Camera on Heterogeneous Parallel Processing Systems. *Seventh International Conference on Complex, Intelligent, and Software Intensive Systems* (pp. 446-450). IEEE.