

# Multiresolution segmentation: a parallel approach for high resolution image segmentation in multicore architectures

Franci Suni Lopez\* y Elisban Flores Quenaya\*

\*Universidad Católica San Pablo, Arequipa, Perú

fran.slom@gmail.com-elisban.flores@gmail.com

**Resumen**—En la interpretación automática de imágenes, el proceso de extracción de los diferentes objetos que componen una imagen es uno de los pasos primarios. Este proceso se conoce como segmentación de imágenes y consiste en subdividir una imagen en regiones significativas, también llamados segmentos, que vayan a ser clasificadas en un paso posterior. Muchos de los algoritmos de segmentación existentes, tienen un alto coste computacional en imágenes de gran tamaño. El objetivo principal de este trabajo es hacer frente a este problema mediante el uso de procesamiento paralelo. La idea es explorar las arquitecturas actuales de varios núcleos disponibles en los procesadores comerciales con el fin de mejorar el tiempo de proceso de segmentación. Nos basamos en la implementación paralela en un algoritmo de regiones crecientes propuesto originalmente por Baatz y Schäpe (2000). Proponemos el método de SLIC-Superpixel para mejorar el tiempo de ejecución, mediante una implementación en CUDA.

**Keywords**—Segmentación de imágenes, procesamiento paralelo, superpixel

## I. INTRODUCCIÓN

En nuestro día a día, realizamos una gran cantidad de tareas de procesamiento de imágenes. Por ejemplo, cuando miramos algo, la primera imagen que nuestros ojos envía al cerebro está posiblemente fuera de foco. El cerebro intenta corregir esto ajustando los lentes oculares, entonces una nueva imagen es enviada de los ojos al cerebro. Este proceso de retroalimentación es tan rápido que no se puede percibir ni sentir[2]. Otro ejemplo es la estereovisión, en donde nuestros ojos envían dos imágenes bidimensionales al cerebro y éste es capaz de fusionarlas en una imagen tridimensional, todo esto de manera instantánea

Más aún, las tareas del procesamiento de imágenes pueden ser parcialmente vistas como un problema de medida, el cual es parte de la ciencia conocida como metrología. Asimismo, las tareas de reconocimiento de patrones son muchas veces incorporadas dentro del procesamiento de imágenes. Existen otras disciplinas con

conexiones relacionadas como: las Redes Neuronales, Inteligencia Artificial y la Percepción Visual [1].

El objetivo de este trabajo es el desarrollo de una aplicación paralela para mejorar los resultados obtenidos por el algoritmo de segmentación de imágenes propuesto en (Baatz et al., 2000) [11] con el método de superpixeles [10]. La idea es aprovechar la capacidad de procesamiento paralelo presente en la mayoría de los procesadores modernos, específicamente en las tarjetas gráficas. Por lo tanto, la solución propuesta requiere de una tarjeta gráfica (nvidia) y puede ejecutarse en máquinas de bajo coste que están disponibles comercialmente. La aplicación paralela se basa en la división del proceso en hilos (procesos). Dado que la calidad de segmentación es un paso crucial para la clasificación, el proceso de parallelización no debe comprometer los resultados. Otra preocupación será mantener el resultado de la segmentación independientemente de la velocidad de ejecución de cada hilo.

En la sección II explicaremos el algoritmo de crecimiento por regiones propuesto por Baatz et al [11], en la sección III se muestra el algoritmo de SLIC-Superpixel. Finalmente en la sección IV y V se presenta los resultados y las conclusiones.

## II. SEGMENTACIÓN POR CRECIMIENTO DE REGIONES

El método es un proceso iterativo de optimización local, lo que minimiza la heterogeneidad promedio de los segmentos generados. La medida de la heterogeneidad utilizado en el algoritmo tiene un componente espacial y un componente espectral. La heterogeneidad espectral se define en los valores de las respuestas espectrales de los píxeles contenidos en un segmento. Esta medida es proporcional a la desviación estándar de la media ponderada para cada banda.

La heterogeneidad espacial se basa en dos atributos de formas: la suavidad y la compacidad. El grado de compacidad se define como la relación entre el

perímetro del segmento y la raíz cuadrada de su área (número de píxeles que contiene). La suavidad se define como la relación entre el perímetro del objeto y el perímetro de la (cuadro delimitador) mínimo rectángulo límite.

Inicialmente, cada segmento representa un solo píxel de la imagen y todos los píxeles están asociados con un cierto segmento. Los segmentos crecen en la medida en que están unidos con sus vecinos, y el menor incremento en la heterogeneidad se utiliza como criterio para la selección de la vecina con la que se adjuntará un segmento. Para simular un crecimiento paralelo, cada segmento se selecciona sólo una vez para cada iteración.

El factor de fusión ( $f$ ) expresa el aumento de la heterogeneidad que resulta de la unión de dos segmentos. Antes de una unión el funcionamiento, el factor de fusión se calcula para cada uno de los vecinos del segmento seleccionado. El vecino que tiene el factor de fusión mínimo se elige para combinación. Sin embargo, la unión sólo se produce si el factor de fusión está bajo cierto umbral, se define como el cuadrado del parámetro de escala, que será denotado en este punto del texto de la letra  $e$ . Este procedimiento continúa la fusión de los segmentos hasta que no haya más candidatos posibles.

El factor de fusión contiene un componente para la heterogeneidad espectral ( $h_{color}$ ) y un componente para la heterogeneidad espacial ( $h_{shape}$ ). La importancia relativa de los componentes espaciales y espectrales se define por el factor de color ( $w_{color}$ ).

$$f = w_{color}.h_{color} + (1 - w_{color}).h_{shape}$$

La siguiente ecuación muestra la formulación de la heterogeneidad espectral; donde el segmento seleccionado es **obj1**, **obj2** es el vecino analizado y el **obj3** es el resultado de la fusión con **obj2** y **obj1**. En esta ecuación  $c$  es el índice de la banda espectral y  $w_c$  es un peso arbitrario establecido para la banda  $c$ ;  $\sigma$  es la desviación estándar de los píxeles de la banda  $c$ , teniendo en cuenta todos los píxeles que pertenecen a **obji** segmento; y  $n$  es el número de píxeles en **obji**, para  $i = 1, 2, 3$ .

$$h_{color} = \sum_i w_c (n_{obj3} \cdot \sigma_c^{obj3} (n_{obj1} \cdot \sigma_c^{obj1} - n_{obj2} \cdot \sigma_c^{obj2}))$$

Las siguientes ecuaciones muestran las formulaciones de los componentes de la compacidad y la suavidad. En estas ecuaciones  $l$  es el perímetro del segmento **obji** y  $b$  el perímetro del cuadro de límite mínimo correspondiente para  $i = 1, 2, 3$ .

$$h_{cmpt} = n_{obj3} \cdot \frac{l_{obj3}}{\sqrt{n_{obj3}}} - (n_{obj1} \cdot \frac{l_{obj1}}{\sqrt{n_{obj1}}}) + n_{obj2} \cdot \frac{l_{obj2}}{\sqrt{n_{obj2}}}$$

$$h_{smooth} = n_{obj3} \cdot \frac{l_{obj3}}{\sqrt{b_{obj3}}} - (n_{obj1} \cdot \frac{l_{obj1}}{\sqrt{b_{obj1}}}) + n_{obj2} \cdot \frac{l_{obj2}}{\sqrt{b_{obj2}}}$$

El crecimiento de los segmentos está limitada, por lo tanto, un criterio ajustable es la heterogeneidad. Este ajuste puede realizarse mediante la elección del parámetro de escala ( $e$ ), los pesos de las bandas espetrales ( $w_c$ ), el factor de color ( $w_{color}$ ) y el factor de compacidad ( $w_{cmpt}$ ). Los cambios en el parámetro de escala influyen directamente en el tamaño de los segmentos generados. Por otra parte, la relevancia de cada banda espectral, la importancia relativa de la forma y color, y entre la compacidad y la suavidad, se pueden ajustar a través de los parámetros del algoritmo.

## II-A. Implementación paralela

La aplicación paralela el algoritmo de región creciente propuesto por Baatz y Schäpe, utiliza la biblioteca de parallelización OpenMP y sigue la división de la informática en los diferentes hilos que comparten la misma área de datos en la memoria. La idea principal de esta solución consiste en dividir la imagen en regiones, que se denota azulejos. Cada baldosa se procesa por un hilo diferente, que realizan una región local en crecimiento, utilizando el algoritmo secuencial, con algunas acciones de sincronización. Este enfoque paralelo, sin embargo, se enfrenta a dos obstáculos principales: (i) el tratamiento de los segmentos de contorno, es decir, segmentos que tienen al menos un vecino que no pertenecen a su azulejo, (ii) la reproducibilidad del resultado final.

En cuanto al tratamiento de los segmentos de delimitación de cada baldosa, la principal dificultad se deriva de los subprocesos que se ejecutan en paralelo. Esto puede hacer que el tratamiento simultáneo del mismo segmento por más de un hilo. Esto podría evitarse con el uso de tramos críticos (zonas en las que sólo un hilo puede ejecutar a la vez) para actualizar el segmento. El uso de las secciones críticas, sin embargo, puede causar un gran impacto en el rendimiento de segmentación, si la contención causada por la espera de secciones críticas es aproximadamente el mismo que los beneficios de la parallelización.

En relación a la reproducibilidad de los resultados, este es un problema inherente al tiempo de ejecución de cada hilo. En otras palabras, un hilo puede realizar su tarea más rápidamente que otros y puede generar diferentes órdenes de las visitas de los segmentos, lo que afecta el resultado final de la segmentación. Incluso para

la segmentación secuencial, si las semillas son visitados en un orden diferente, el resultado de la segmentación se modifica. La reproducibilidad del resultado de la segmentación, sin embargo, es un objetivo importante, ya que permite a los científicos desde diferentes ubicaciones para generar segmentaciones de la misma imagen, y por lo tanto miran el mismo resultado.

Para que el proceso de segmentación sea realmente independiente de la velocidad de los hilos y para evitar la contención excesiva de las secciones críticas, los segmentos situados en los límites de las baldosas se tratan por separado en el algoritmo. Estos segmentos, llamados a partir de ahora en los segmentos de frontera, se incluyen en una lista de segmentos a tratar. Al final de cada paso de la segmentación (después de todos los segmentos han sido visitados), los segmentos de frontera serán procesadas secuencialmente. Por lo tanto, el crecimiento de las regiones de cada hilo será independiente, sin necesidad de las secciones críticas en el código.

La división de la imagen en azulejos, y en consecuencia la división del trabajo en las discusiones, pueden afectar el resultado final de la segmentación. Para lograr un mejor rendimiento en una determinada arquitectura con múltiples núcleos, lo ideal es que el número de hilos es siempre igual al número de núcleos de procesador disponibles. En nuestra aplicación, el usuario define el número de subprocesos que se ejecutan en el procesador. Esto garantiza la misma división baldosa y la reproducibilidad de los resultados de la segmentación para diferentes arquitecturas.

*1) Distribución inicial:* La primera etapa del algoritmo consiste en la determinación de la número de baldosas que se generará. El número de azulejos corresponde a la cantidad de hilos. Si sólo existe un hilo, la cálculo es secuencial. Para dos o más hilos, la imagen es dividida en áreas distintas, como se muestra en la Figura 1. Cada hilo está responsable del tratamiento de los píxeles incluidos en su mosaico.

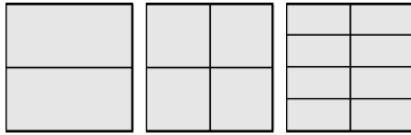


Figura 1: División de baldosas para 2, 4 y 8 threads.

*2) Crecimiento de regiones:* Despues de la división inicial de la imagen, comienza comienza la etapa de crecimiento. Cada hilo ejecuta el algoritmo de crecimiento dentro de su propia región de baldosas. Inicialmente, el hilo marca todos los píxeles de la baldosa como segmentos a ser visitados y los organiza en una lista de segmentos. Los segmentos se incluyen en esta lista

en el mismo orden en que serán visitados. Con el fin de iniciar el crecimiento de los segmentos relativamente distantes, los segmentos se incluyen en función de su distancia relativa en la imagen.

Un hilo visita cada segmento de su lista, y analiza el aumento de heterogeneidad para cada uno de sus vecinos. Si al menos uno de los vecinos no pertenece a la baldosa tratada por dicho hilo, el segmento se incluye en la lista de segmentos de frontera. De lo contrario, el segmento se procesa normalmente y marcado como visitado. El vecino que resulte en la menor aumento de la heterogeneidad es considerado el mejor vecino. Si se considera a este vecino como el mejor, por el factor de fusión, como parte del segmento, a continuación, se produce una fusión. Este procedimiento se repite hasta que toda la lista de segmentos en cada hilo está cubierto.

Después de que todos los hilos terminen su cómputo, los segmentos de frontera se manejan. La lista de segmentos de frontera es atravesada secuencialmente por un único hilo, utilizando el mismo algoritmo región de cultivo. La lista de segmentos de frontera se visitó de manera de intercalación, un segmento de frontera de cada baldosa en el tiempo. Por lo tanto, los segmentos permanecen visitaron de una manera distribuida.

Después de la lista de segmentos de frontera se procesa, la etapa de crecimiento se ha completado, y el algoritmo se inicia una nueva etapa, con otra etapa de crecimiento. La nueva etapa de crecimiento se inicia de la misma manera, con un número de hilos de computación los segmentos de cada azulejo. En esta etapa, sin embargo, la lista de segmentos de cada hilo se compone de los segmentos generados en el paso anterior. Este proceso se repite, la generación de nuevos pasos, hasta que no se mezcla en un paso o hasta que se alcanza un número máximo de pasos.

### III. SLIC SUPERPIXEL SEGMENTATION

La representación de una imagen por medio de píxeles es en muchas ocasiones, redundante ya que los objetos de interés están compuestos por multitud de píxeles similares, esto lleva a malgastar recursos computacionales[12]. Los superpixels son regiones contiguas de una imagen perceptualmente similares. Idealmente, todos los píxeles dentro de un mismo superpixel, pertenecen al mismo objeto del mundo real. La segmentación de imágenes utilizando superpixels tiene el potencial de reducir de manera notable el coste del análisis automático de imágenes, ya que disminuye el número de elementos analizados por imagen, de miles de píxeles a cientos de superpixels [13]. Además, los superpixels pueden incorporar, implícitamente, información de la forma del objeto, y nos delimitan qué píxeles debemos procesar conjuntamente.

### III-A. Algoritmo

SLIC es simple de usar y entender. De manera predeterminada, el único parámetro del algoritmo es  $k$ , el número deseado de superpíxeles, aproximadamente del mismo tamaño. El procedimiento de clustering comienza con una etapa de inicialización donde los  $k$  centros de los clusters iniciales  $C_i = [l_i, a_i, b_i, x_i, y_i]^T$  se muestran en una cuadrícula regular espaciados por  $S$  píxeles de distancia. Para producir superpíxeles más o menos del mismo tamaño, el intervalo de cuadrícula es  $S = \sqrt{N/k}$ . Los centros se mueven al lugar de las semillas, correspondiente a la posición del píxel con más baja gradiente en un espacio de  $3 \times 3$ . Esto se hace para evitar centrar un superpíxel en un borde y para reducir la posibilidad de sembrar un superpíxel con un pixel ruidoso.

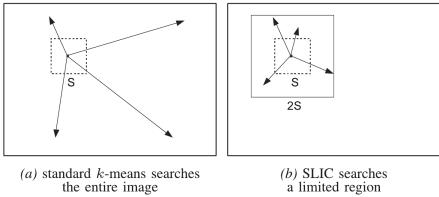


Figura 2: Asignación de cada píxel.

A continuación, en la etapa de asignación, cada pixel  $i$  está asociado con el centro de la agrupación más cercana cuya región de búsqueda se superpone su ubicación, tal como se representa en la Figura 2. Esta es la clave para la aceleración del algoritmo porque la limitación del tamaño de la región de búsqueda reduce significativamente el número de cálculos de distancia, y resulta en una ventaja significativa sobre la velocidad convencional de k-means clusters donde cada píxel debe ser comparado con todos los centros de los clusters. Esto sólo es posible a través de la introducción de una medida de la distancia  $D$ , que determina la agrupación centro más cercano para cada píxel. Dado que la extensión espacial esperada de un superpíxel es una región del tamaño aproximado  $S \times S$ , la búsqueda de los píxeles similares se lleva a cabo en una región  $2S \times 2S$  alrededor del centro del superpíxel.

Una vez que cada píxel se ha asociado a la agrupación centro más cercano, una etapa de actualización ajusta los centros de los clusters. La norma L2 se utiliza para calcular un error de E residual entre las nuevas ubicaciones de los centros de clúster y ubicaciones de los centros de clúster anteriores. Los pasos de asignación y actualización se pueden repetir de forma iterativa hasta que converge el error, pero hemos encontrado que 10 iteraciones es suficiente para la mayoría de las imágenes, e informar de todos los resultados en este trabajo utilizando este criterio. Por último, una etapa de

postprocesado hace cumplir la conectividad mediante la reasignación de píxeles disjuntos superpíxeles cercanas. Todo el algoritmo se resume en el algoritmo 1.

---

#### Algorithm 1 SLIC superpixel segmentation

---

```

1: /* Initialization */
2: Initialize cluster centers  $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ 
   by sampling pixels at regular grid steps S.
3: Move cluster centers to the lowest gradient position
   in a  $3 \times 3$  neighborhood.
4: Set label  $l(i) = -1$  for each pixel i.
5: Set distance  $d(i) = \infty$  for each pixel i.
6: repeat
7:   /* Assignment */
8:   for each cluster center  $C_k$  do
9:     for each pixel i in a  $2S \times 2S$  region around
    $C_k$  do
10:    Compute the distance D between  $C_k$  and
    i.
11:    if  $D < d(i)$  then
12:      set  $d(i) = D$ 
13:      set  $l(i) = k$ 
14:    end if
15:   end for
16: end for
17: /* Update */
18: Compute new cluster centers.
19: Compute residual error E.
20: until  $E \leq \text{threshold}$ 

```

---

## IV. RESULTADOS

En esta sección, se presentan los resultados del artículo original, nuestra implementación en OpenMP y la implementación en CUDA. Las siguientes secciones describen el entorno utilizado en los experimentos, las imágenes de prueba, y los resultados de segmentación, junto con la evaluación de los resultados obtenidos con la parallelización.

### IV-A. Entorno de prueba

Los experimentos fueron realizados en un procesador Intel Core i7-4770M CPU @ 3.40GHz x8 con 8 GB de RAM.

Se utilizaron tres imágenes con diferentes tamaños y características. Se nombran como Im1, Im2 y Im3 y se exponen, respectivamente, en las figuras 3, 4 y 5. Los tamaños de imagen se presentan en la Tabla I. Las tres imágenes se utilizaron para evaluar las mejoras de rendimiento y para comparar el resultado generado a partir de la segmentación paralelo al resultado de la segmentación secuencial.



Figura 3: Im1.

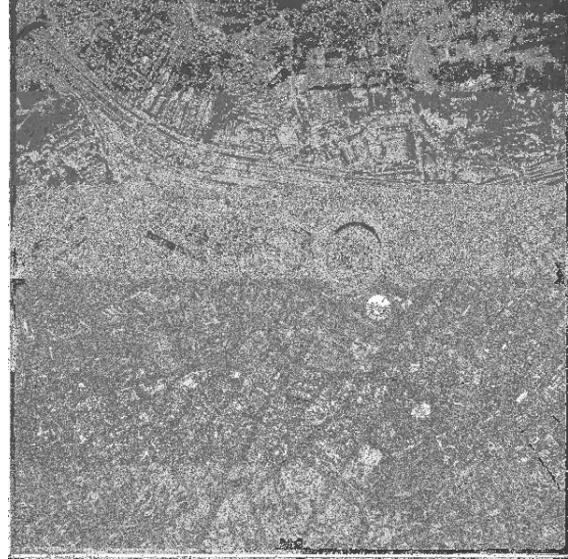


Figura 5: Im3.



Figura 4: Im2.

Imagen	Tamaño (pixels)
Im1	1000x1000
Im2	2000x2000
Im3	2800x2800

Cuadro I: Tamaño de las imágenes usadas

La Tabla II presenta el tiempo de ejecución de la segmentación del artículo original con el número de hilos ejecutados. La tabla III muestra los tiempos de ejecución de la segmentación con nuestra implementación en OpenMP y CUDA, con la misma cantidad de hilos. Tenga en cuenta que el tiempo de ejecución de la segmentación se reduce con el aumento en el número de hilos.

Imagen	Tiempo (seg)		
	1 thread	2 threads	4 threads
Im1	13.67	8	6.33
Im2	62.33	39.33	24
Im3	123.66	76	51

Cuadro II: Tiempos de ejecución artículo original.

#### IV-B. Resultados de segmentación

En las figuras 6, 7 y 8, se muestra los resultados del proceso: la imagen original (a), la imagen procesada por el algoritmo de crecimiento de regiones [11] (b), nuestra implementación de SLIC-Superpixel con OpenMP (c) y SLIC-Superpixel con CUDA (d).

#### IV-C. Evaluación del desempeño

El rendimiento de los algoritmos propuestos, han sido evaluados utilizando las tres imágenes, variando el número de hilos y la implementación en CUDA.

Imagen	Tiempo (seg)				
	1 thread	2 threads	4 threads	8 threads	CUDA
Im1	5.82	3.06	1.62	1.4	0.79
Im2	22.38	11.6	6.28	5.26	2.84
Im3	44.56	17.25	14.4	10.15	5.4

Cuadro III: Tiempos de ejecución con OpenMP y CUDA.

Los resultados sugieren una reducción aún mayor de tiempo segmentación si se utilizan más procesadores. Y

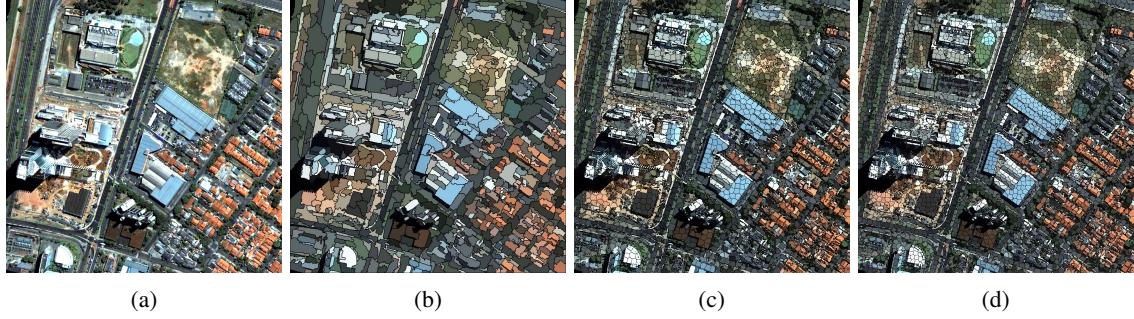


Figura 6: Resultados **Im1**. (a) Imagen original. (b) Baatz y Schäpe algoritmo. (c) OpenMP. (d) CUDA



Figura 7: Resultados **Im2**. (a) Imagen original. (b) Baatz y Schäpe algoritmo. (c) OpenMP. (d) CUDA

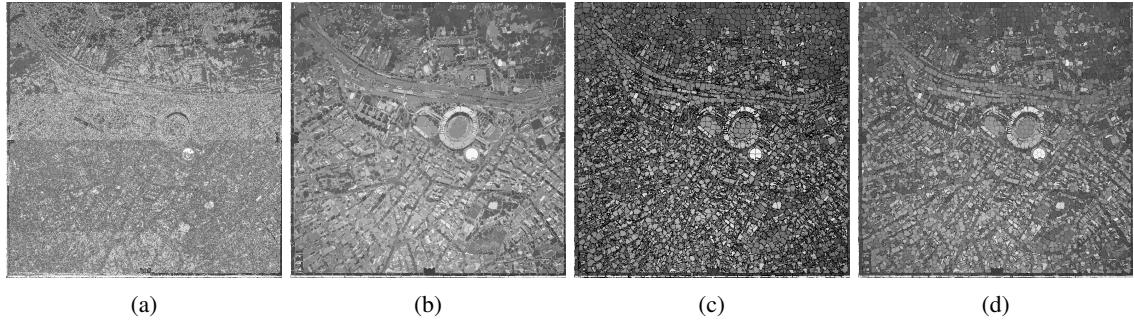


Figura 8: Resultados **Im3**. (a) Imagen original. (b) Baatz y Schäpe algoritmo. (c) OpenMP. (d) CUDA

al un tiempo menor con la implementación en CUDA. Como se conoce, las implementaciones en CUDA consideran un tiempo de ejecución en el DEVICE y un tiempo de copiado de las variables, en la tabla IV, se muestra a detalle los tiempos de ejecución y copiado de cada imagen.

Imagen	Tiempo (seg)		
	Copiado	Cálculo DEVICE	Total
Im1	0.07656	0.71779	0.79435
Im2	0.28221	2.55304	2.83525
Im3	0.54683	4.90115	5.44798

Cuadro IV: Tiempos de ejecución y copiado en CUDA.

## V. CONCLUSIONES

Este trabajo presenta el algoritmo de crecimiento de regiones propuesto por Baatz y Schäpe, la implementa-

ción de la técnica de SLIC-Superpixel paralelizado en OpenMP y CUDA.

Como es evidente al paralelizar el algoritmo el tiempo de ejecución se reduce y el cambio se nota más con la implementación en CUDA. También concluimos que al ser los datos pequeños el tiempo de copiado del HOST al DEVICE es mínimo (90 % del tiempo de ejecución en el DEVICE y 10 % en el copiado).

Además, nuestra implementación cumple el trabajo futuro del artículo original; de utilizar una técnica GPU (Graphics Processing Units), para acelerar más el proceso de segmentación.

## REFERENCIAS

- [1] M.A. Aguilar, M.M. Saldaña, F.J. Aguilar *GeoEye-1 and WorldView-2 pan-sharpened imagery for object-based classification in urban environments* Int. J. Remote Sens., 34 (7) (2012), pp. 2583–2606
- [2] J.P. Ardila, W. Bijker, V.A. Tolpekin, A. Stein *Context-sensitive extraction of tree crown objects in urban areas using VHR satellite images* Int. J. Appl. Earth Obs. Geoinf., 15 (2012), pp. 57–69
- [3] N. Clinton, A. Holt, J. Scarborough, L. Yan, P. Gong *Accuracy assessment measures for object-based image segmentation goodness* Photogramm. Eng. Remote Sen., 76 (3) (2010), pp. 289–299
- [4] D.C. Duro, S.E. Franklin, M.G. Dubé *A comparison of pixel-based and object-based image analysis with selected machine learning algorithms for the classification of agricultural landscapes using SPOT-5 HRG imagery* Remote Sens. Environ., 118 (2012), pp. 259–272
- [5] Y. Gao, J.F. Mas, N. Kerle, J.A. Navarrete Pacheco *Optimal region growing segmentation and its effect on classification accuracy* Int. J. Remote Sens., 32 (13) (2011), pp. 3747–3763
- [6] C.A. Laben, B.V. Brower *Process for Enhancing the Spatial Resolution of Multispectral Imagery Using Pan-Sharpening* Eastman Kodak, US Patent (2000) (6,011,875)
- [7] T.R. Martha, N. Kerle, C.J. van Westen, V. Jetten, K.V. Kumar *Segment optimization and data-driven thresholding for knowledge-based landslide detection by object-based image analysis* IEEE Trans. Geosci. Remote Sens., 49 (12) (2011), pp. 4928–4943
- [8] H. Zhang, J. Fritts, S. Goldman *Image segmentation evaluation: a survey of unsupervised methods* Comput. Vis. Image Underst., 110 (2) (2008), pp. 260–280
- [9] Baatz, M. and Schape, A. *“Multiresolution segmentation: an optimization approach for high quality multi-scale image segmentation”*. In: XII Angewandte Geographische Informationsverarbeitung, Wichmann-Verlag, Heidelberg, 2000.
- [10] Engel, D.; Spinello, L.; Triebel, R.; Siegwart, R.; Bulthoff, H.H.; Curio, C. *Medial Features for Superpixel Segmentation* Proceedings of the Eleventh IAPR Conference on Machine Vision Applications (MVA 2009)
- [11] P. N. Happ, R. S. Ferreira, C. Bentes , G. A. O. P. Costa, R. Q. Feitosa *Multiresolution segmentation: a parallel approach for high resolution image segmentation in multicore architectures* Conference: International Conference on Geographic Object-Based Image Analysis (GEOBIA 2010), At Ghent, Belgium
- [12] A. P. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones, “*Superpixel lattices*,” in CVPR, 2008.
- [13] F. Drucker and J. MacCormick, “*Fast superpixels for video analysis*,” in Proceedings of the international conference on Motion and video computing, 2009.