# Accelerated gSLIC for Superpixel Generation used in Object Segmentation

## 1. Alcance del articulo.

El objetivo del artículo a implementar **[1]** es crear una segmentación robusta de objetos. Se utiliza un método basado en segmentación por super-píxeles el cual debe ser capaz de ejecutarse en aplicaciones en tiempo real.

El algoritmo **SLIC** propuesto por Achanta et al. **[2]** es un algoritmo de segmentación de super-píxeles basado en la agrupación k-medias. Las aplicaciones que utilizan super-píxeles han sido propuestas por los autores de la **gSLIC**, la cual es la versión en GPU del algoritmo **SLIC**.

Los autores de **[1]** explican que el algoritmo de segmentación de imagen **gSLIC** consta de dos partes principales, el clustering y la conectividad, ellos aceleraron exitosamente la parte de clustering, pero la parte de "hacer cumplir la conectividad de la etiqueta" no es paralelizable, al menos no para la arquitectura CUDA ya que este paso del algoritmo es recursivo, realizaron pruebas basadas en procesamiento morfológico, se logró una pequeña aceleración, pero fue a costa de la calidad de la segmentación. Como trabajo futuro desean continuar en esta tarea para lograr la aceleración deseada sin ninguna degradación de la calidad.

## 2. Implementación en CUDA

NVIDIA creo el Amazon Machine Image con CUDA Toolkit 7.5 en el sistema operativo Amazon Linux 2016.03 (arquitectura de 64 bits).
El kit de herramientas CUDA proporciona un entorno de desarrollo para desarrolladores de C / C ++ / Python que crean aplicaciones aceleradas por GPU. Incluye un compilador para GPU NVIDIA, bibliotecas de matemáticas y herramientas para depurar y optimizar el rendimiento de aplicaciones.
Es por eso que se opta por alquilar un servidor EC2 de la nube de Amazon, el cual, tiene un tarjeta de video con 3072 núcleos de procesamiento.
Se opta por utilizar Python con la librería Numba para explorar las facilidades de implementación que nos proporciona Numba.

El algoritmo de gSLIC implementado es el siguiente:

**gSLIC(imagen, m, S, threshold)**

1. Transformar la imagen del espacio de color rgb al espacio CIELAB
2. Inicializar Centros de super-píxeles
3. Realizar la clusterización.
    a. Copiar a la GPU la imagen en CIELAB
    b. Copiar a la GPU el arreglo de centros creado en el paso 2
    c. Copiar memoria en GPU para la matriz de etiquetas de centro
    d. Crear la configuración de ejecución del kernel de clusterización con un arreglo de hilos por bloque de 32 x 32
    e. Crear la configuración de ejecución del kernel de clusterización con un arreglo de bloques de (ancho imagen/32, alto de imagen/32)
    f. Ejecutar el kernel de clusterización
    g. Copiar la matriz de etiquetas de centro a la CPU
    h. Calcular los nuevos centros y el error global basado en la distancia L1.
    i. Si el error global es menor o igual al Threshold salir, caso contrario ir al paso 3c.
4. Ejecutar conectividad.

El kernel de clusterización es el siguiente:

**Kernelclustering (imagenLab, centros, etiquetas, m, S):**

1. Calcular las coordenadas x, y de los hilos y bloques sobre la imagen lab.
2. Iterar sobre todos los centros.
    a. Si el píxel pertenece al vecindario 2S * 2S del centro calcular distancia a este, si la distancia es la menor de entre todas (hacia los otros centros), etiquetar a este píxel como perteneciente a este centro.

## 3. Resultados obtenidos.

La segmentación del algoritmo **gSLIC** genera super-píxeles compactos sobre una imagen en color, en la **Figura 1** observamos una imagen que ha sido segmentada utilizando el algoritmo propuesto en **[1]** y la implementación usando Numba que se ha propuesto en este documento técnico.

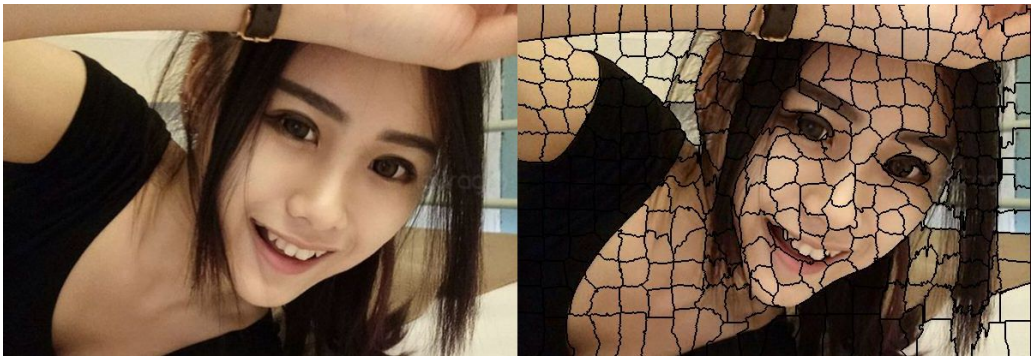


**Figura 1. Segmentación de imagen de color con la implementación propuesta, los parámetros aplicados sobre el algoritmo son m = 20, S = 30 y Threshold = 0.1**

El parámetro “m” indica la compactación del superpixel según el artículo original en **[2]** los valores para m van desde 1 a 20, el parámetro S nos indica el tamaño inicial de los super-píxeles, S*S píxeles y el threshold indica hasta cuando deben de iterar los super-píxeles para encontrar el centro correcto.

Se observa que al disminuir el threshold la calidad de los super-píxeles mejora como se observa en la **Figura 2**, pero en la implementación en CPU este cambio conlleva màs tiempo de procesamiento.

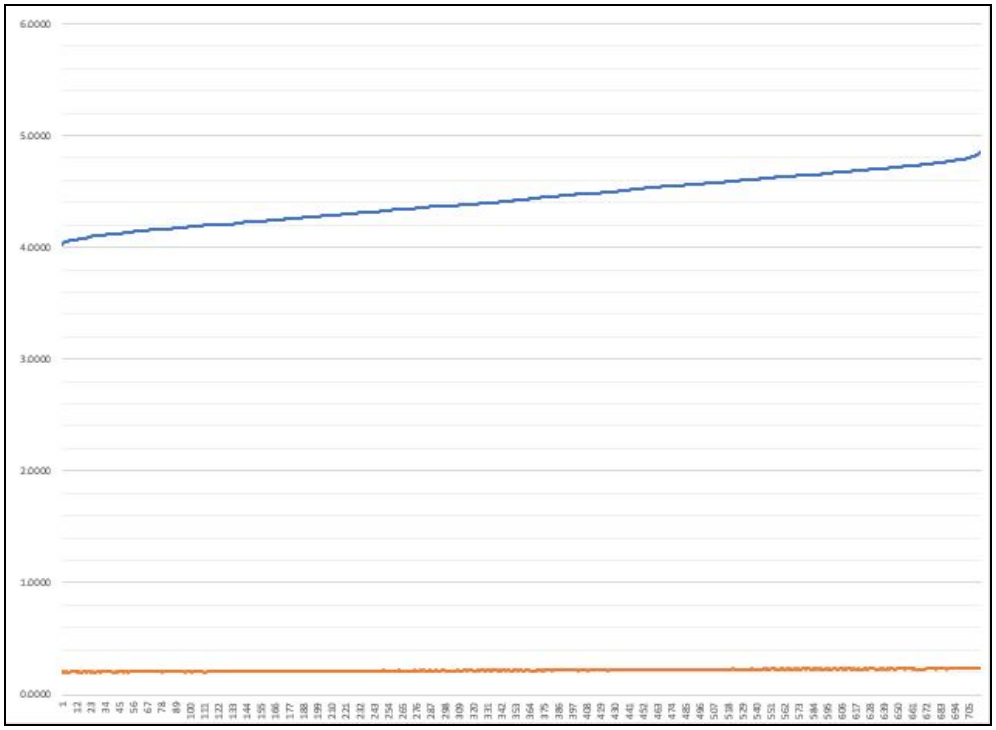
**Figura 2. Segmentación de imagen de color con la implementación propuesta, los parámetros aplicados sobre el algoritmo son m = 20, S = 20 y Threshold = 0.05**

En la **Figura 3** se observa una degradación en la calidad de los super-píxeles generados esto es debido a que el parámetro m està establecido a un valor de 1.

**Figura 3. Segmentación de imagen de color con la implementación propuesta, los parámetros aplicados sobre el algoritmo son m = 1, S = 20 y Threshold = 0.05**

En la **Figura 4**. Aumentamos el tamaño del super-pixel a 40, se observa que hay super-pixeles que reducen su tamaño y otros que aumentan

**Figura 4. Segmentación de imagen de color con la implementación propuesta, los parámetros aplicados sobre el algoritmo son m = 20, S = 40 y Threshold = 0.05**




**Cuadro 1. Comparativa entre la ejecución del algoritmo SLIC y gSLIC**

## 4. Bibliografia.

**[1]** Birkus, R. (2015). Accelerated gSLIC for superpixel generation used in object segmentation. *Proc. of Center of Excellence in Stem Cell Genomics*.

**[2]** *Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. SLIC Superpixels. Technical report, EPFL, 2010.*