Berkeley Segmentation Data Set and Benchmarks 500

Silvana Castillo
Universidad de los Andes
Cra 1 #18a-12
ls.castillo332@uniandes.edu.co

Laura Daza
Universidad de los Andes
Cra 1 #18a-12
la.daza10@uniandes.edu.co

Abstract

La segmentación de imágenes es una de las bases principales para diversas aplicaciones de la vision por computador. En este articulo se comparan los 5 métodos básicos de segmentación: k-means, gmm, hierarchical y watersheds; en diferentes espacios de color y teniendo (o no) en cuenta la información espacial. Se analiza, el resultado de las mejores combinaciones "hierarchical-RGBz "K-means-Lab"(con varios k) evaluando su desempeño en BSDS500 con el fin de obtener valores comparables entre métodos (curva de precisión vs recall), con el estado del arte y el desempeño humano. Adicionalmente, proponen ajustes para mejorar los resultados del algoritmo.

1.. Introducción

La segmentación de imágenes continua siendo uno de los principales bloques de construcción en muchas aplicaciones diferentes de visión por computador. De manera general, la segmentación tiene el objetivo de dividir la imagen en un conjunto de secciones de interés basándose en la similaridad entre los objetos que componen cada uno de ellos. Algunos de los principales métodos que existen para llevar acabo este proceso se encuentran los siguientes:

1.1.. K-means

Es uno de los métodos mas simples de aprendizaje no-supervisado, se encarga de separar los elementos de una imagen en k grupos diferentes basado en la similitud entre los píxeles. Ahora, este método utiliza la distancia euclidiana para ver que tanto se parecen los píxeles a un centroide inicializado al azar utilizando sus distintas características como color, textura o ubicación espacial. [2] [3]

1.2.. Gmm

Es una generalización de K-means en la cual los centroides, también inicializados de forma aleatoria, ya no son un punto sino una gaussiana, lo cual resulta en que a cada píxel no se le es asignado un grupo especifico sino una serie de probabilidades de pertenecer a distintos grupos basado en la distancia al centro de cada uno de ellos. [2]

1.3.. Hierarchical

Este tipo de segmentación calcula la distancia entre los píxeles de la imagen y agrupa los que estén mas cercanos. Después de esto re-calcula las distancias tomando ahora el nuevo grupo como un solo elemento y vuelve a agrupar los mas cercanos. Este procedimiento es iterado hasta que se obtiene un solo grupo y los clusters son obtenidos umbralizando el dendograma resultante de las asociaciones. [2] Este método para segmentar imágenes es bastante flexible ya que permite controlar que tipo de distancia se desea calcular y como calcularla entre los grupos recién formados, sin embargo es costoso computacionalmente por lo cual es demorado cuando el problema es grande.

1.4.. Watershed

Es un método de segmentación que utiliza los mínimos locales para crear clusters a partir de ellos utilizando el concepto geográfico de la linea divisoria de aguas, en el cual los grupos son zonas en donde cuando caiga una gota de agua siempre se dirigirá hacia el mismo punto: la zona mas profunda de la región. Pero una imagen es una superficie muy irregular lo que causa un alto numero de mínimos locales resultando en una sobre-segmentación. Aun así, el método es bastante poderoso si se realiza un proceso de imposición de mínimos en la imagen ya que así es posible controlar el número de clusters final. [2]

2.. Métodos de segmentación

Se creo una función de segmentación en Matlab con la siguiente signatura y especificaciones para cada parámetro:

```
function my_segmentation = segment_by_clustering(
    rgb_image, feature_space, clustering method,
    number of clusters)
```

- **rgb_image:** La imagen (matriz) que se quiere segmentar en formato rgb.
- **feature_space:** El espacio de color (y si se le da importancia a la posición de cada pixel) que se quiere tener en cuenta para realizar la segmentación. Con las siguientes opciones: rgb, lab, hsv, rgb+xy, lab+xy o hsv+xy.
- **Clustering_method:** El tipo de método de agrupamiento a usar, con las siguientes opciones: k-means, gmm, hierarchical o watershed.
- **Number of clusters:** El valor k para los métodos que lo necesiten, cantidad de cluster en los que se va a segmentar la imagen; si no se pone ninguno se asume que el valor es 5.

El resultado de esta función es una imagen (o matriz) donde cada pixel tiene una etiqueta que corresponde a uno de los cluster. Es decir, dos pixeles que pertenecen al mismo cluster deben tener la misma etiqueta.

2.1.. Métodos escogidos

Entre todas las opciones, se eligieron las combinaciones:

- Hierarchical en RGB
- K-means en Lab

Se realizaron diversos experimentos comparativos, como los que se pueden apreciar en las figuras 1, 2, 3 y 4, con el fin de encontrar las dos combinaciones que dieran mejores resultados visualmente.

Inicialmente se descarto el método de Watersheds debido a que los resultados visuales en varias imágenes de prueba eran los peores entre todos los métodos, como se puede comprobar en los ejemplos. De la misma manera fue descartado el canal de color HSV, ya que en la mayoría de imágenes de prueba se puede ver que falla, ya sea que segmentaba toda la imagen como un solo objeto o no obtenía regiones de interés y/o coherentes. Entre las combinaciones restantes se evaluó cualitativamente la segmentación utilizando el espacio de color RGB y se encontró que de estos grupos el que retornaba mejores resultados es hierarchical sin considerar las coordenadas espaciales. Por otro lado, cuando se considero el espacio de color Lab el mejor método generalmente era k-means, también sin tener en cuenta la información espacial. La información espacial fue descartada debido a que en varios resultados, se puede ver que la posición no es tan relevante para las imágenes y tenerla en cuenta solo lograba que se segmentaran regiones incoherentes de tamaño similar que no aportaban información relevante de la imagen.

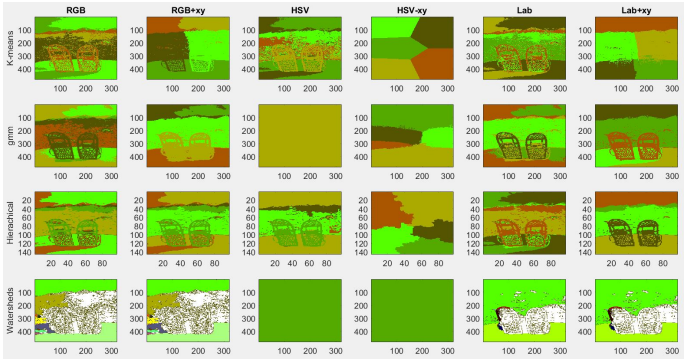

Figura 1. Resultado de todos los métodos en cada uno de los espacios de color disponibles con k=5, en la imagen 2018.jpg

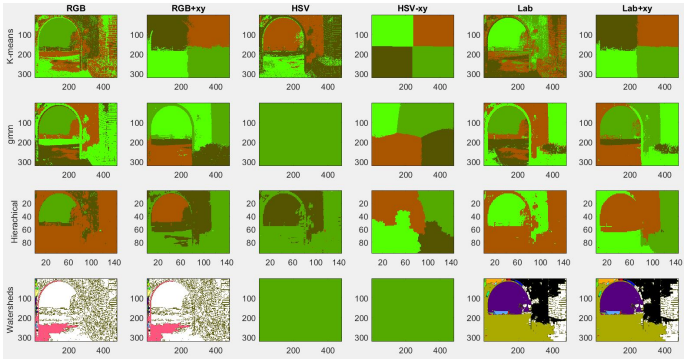

Figura 2. Resultado de todos los métodos en cada uno de los espacios de color disponibles con k=4, en la imagen 5096.jpg

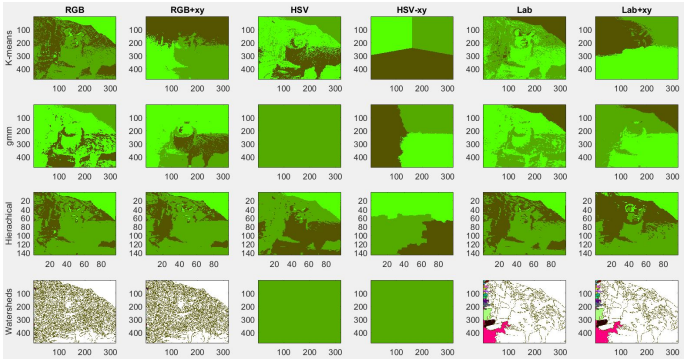

Figura 3. Resultado de todos los métodos en cada uno de los espacios de color disponibles con k=3, en la imagen 6046.jpg

3.. Metodología de pruebas

El objetivo del Benchmark era producir una puntuación confiable y reproducirle para un algoritmo de segmentación o detección de contornos, esto con el fin de poder comparar dos algoritmos diferentes y para poder obtener un seguimiento del avance de estas técnicas hacia el rendimiento a nivel humano.[1]

La base de datos BSDS500 es una extensión de BSDS300, donde las 300 imágenes originales se usan pa-

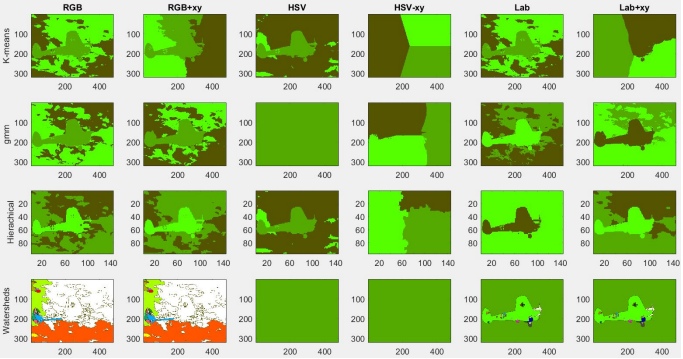

Figura 4. Resultado de todos los métodos en cada uno de los espacios de color disponibles con k=3, en la imagen 3063.jpg

ra la parte de entrenamiento y validación y 200 imágenes nuevas, con anotaciones, fueron agregadas para la parte de evaluación. Cada imagen tiene un ground-truth que consiste de un archivo *.mat* que tiene la segmentación de la imagen dada por 5 personas diferentes; la mitad de las segmentaciones se obtuvieron presentando a los sujetos la imagen a color, y la otra mitad presentándola en escala de grises. Las imágenes están en formato jpg y tienen un tamaño de 321x481 píxeles.[1]

La base de datos se encuentra dividida en subconjuntos de entrenamiento, validación y evaluación. Con el fin de permitir obtener un resultado comparable con otros métodos se debe realizar toda la parte de aprendizaje, ajustes de parámetros, selección de modelo y demás, en los subconjuntos de entrenamiento y validación. Después del entrenamiento, el algoritmo solo se debe correr una vez con los parámetros fijos, es decir no se debe usar el subconjunto de test ni su ground-truth para ajustar los parámetros del algoritmo.[1]

4.. Resultados

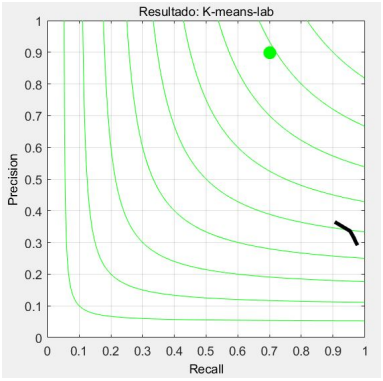

Figura 5. Curva de precisión vs recall usando k-means-Lab usando 200 imágenes

Se segmentaron todas las imágenes del test usando la combinación k-means-Lab, con incrementos de k de 10 en 10, creando para cada imagen 5 segmentaciones. De esta manera, se obtuvo el resultado de la figura 5.

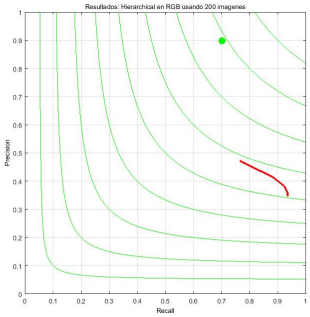

Figura 6. Curva de precisión vs recall usando hirarchicals-RGB (izquierda) usando 200 imágenes

Se realizo la combinación hierachicals-rgb en todas las imágenes del test, creando para cada imagen 5 segmentaciones diferentes con variaciones de k de 5 en 5. De esta manera, usando las 200 imágenes se obtuvo el resultado mostrado en la figura 6.

Se intento obtener las gráficas múltiples veces buscando mejorar los resultados obtenidos (alargar la gráfica). Para esto, se intento agrandar el rango de k, aumentar cantidad de segmentaciones, disminuirlas y finalmente disminuir cantidad de imágenes. Finalmente al disminuir la cantidad de imágenes, se adquirió el resultado de la figura 7 usando solamente los archivos *.mat* de las primeras 5 imágenes, este procedimiento se repitió duplicando la cantidad de imágenes y hasta 50 el resultado es el mismo; pero si se aumenta mas da nuevamente el mismo resultado de la figura 6.

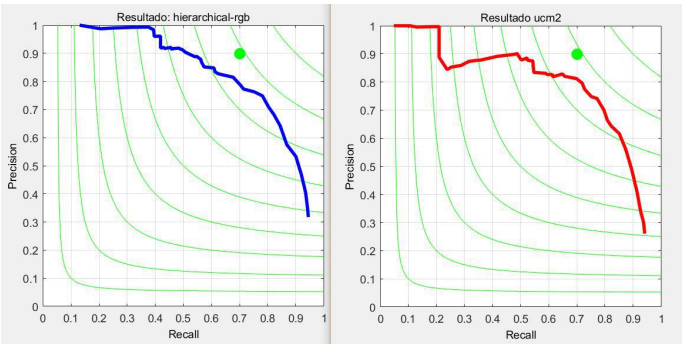

Figura 7. Curva de precisión vs recall usando hirarchicals-RGB (izquierda) y usando los resultados precalculados de *Ultrametric Contour Map* (con solo 5 imágenes)

5.. Discusión

Como se menciono en los resultados, intentamos diferentes ajustes con el fin de mejorar los resultados, por sentido común se asumió que aumentando los paramentos (k,

cantidad de segmentaciones, cantidad de imágenes) aumentaría el alcance de la gráfica ya que permitiría evaluar el algoritmo a diferentes umbrales, sin embargo descubrimos que era todo lo contrario, al disminuir los parámetros, la curva resultante abarcaba un rango mayor tanto de recall como de precisión; creemos que esto se debe a que permite evaluar solo las imágenes con mejores resultados.

Con base en el resultado de la combinación k-means-lab, se puede ver que la curva obtenida es bastante corta, por lo que no es posible realizar un análisis exhaustivo. No obstante, de la gráfica se puede rescatar que el método obtiene valores de recall bastante altos (casi llegando a 1), y teniendo en cuenta la inclinación de la curva se puede inferir que el desempeño del algoritmo tiende a una precisión mayor probablemente alcanzando 0.6 aproximadamente.

Tomando únicamente el resultado obtenido en la figura 6, no se obtuvo una curva completa que permita un mejor análisis. Sin embargo, podemos ver que la combinación hierarchicals-RGB (con todas las imágenes) tiene un recall bastante alto (casi alcanzando 0.95) y que a medida que este baja su precisión va aumentando (casi alcanzando 0.5), si se realiza una extrapolacion siguiendo la trayectoria que muestra la curva, se puede asumir que la mayor precisión que conseguirá el algoritmo sera de aproximadamente 0.7 (lo cual es un resultado bastante bueno par un algoritmo tan generalizado).

Por otro lado, debido a que las curvas resultantes iniciales no eran lo esperado, salían muy cortas e incompletas, el resultado de la figura 7 no parece bastante fiable, por lo tanto se decidió realizar una curva adicional con los archivos *.mat* precalculados de *Ultrametric Contour Map* (UCM2 que se puede visualizar en la figura 7) que ya venían inicialmente con el código de BSDS, con el fin de compararlas para ver si el resultado era coherente o en algún momento se gráfico lo que no era.

Si se tienen en cuenta estos resultados, se puede ver que desempeño entre hierarchicals-RGB y UCM2 es bastante similar. Nuestro método muestra una mayor consistencia y una precisión casi perfecta en un rango mas grande de recall (0.15-0.4), no obstante UCM2 consigue una precisión muy alta en rangos de recall a los que no llega hierarchicals-RGB (por debajo de 1.5) y se acerca mas al desempeño humano (a simple vista parece tener un valor de Fmax mayor).

6.. Conclusiones

Debido a que el algoritmo de segmentación evaluado se realizó usando solamente 5 imágenes, es decir que se entrenó solo con estas, el principal obstáculo al que nos enfrentábamos era el riesgo de hacer un método especifico para esas pocas imágenes (overfitting). Es por esa razón que se tomo la decisión de realizar un algoritmo poco complejo para prevenir que eso suceda, pero esto resultó en que su rendimiento fuera insuficiente. Sin embargo, la utilización de las 300 imágenes de la base de datos de Berkeley ayudaría a la corrección de este inconveniente debido a que permite analizar su funcionamiento en distintos tipos de imágenes y por consiguiente posibilita la adecuación del algoritmo a esos cambios sin hacer que el código sea especifico para una imagen o un pequeño grupo de estas.

Por otro lado, la variación de los pesos entre la información de color y la información espacial utilizando diferentes combinaciones permitiría eventualmente encontrar una que optimice el rendimiento al tener mas en cuenta las variables que mejor describan a los elementos. Adicionalmente, se podrían utilizar otros descriptores de la imagen como textones que brindarían mas información para separar mejor los píxeles por categorías.

Referencias

[1] Arbelaez, P. *The Berkeley Segmentation Dataset and Benchmark*. Contour Detection and Image Segmentation Resources. Recuperado 17 de marzo de 2016de: `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/`

[2] Arbelaez, P. et al. *Notas de clase: Segmentación*. Visión por computador. Ingeniería Biomedica - Universidad de los Andes.

[3] MATLAB & Simulink, *k-Means and k-Medoids Clustering*. Recuperado 17 de marzo de 2016de: `http://www.mathworks.com/help/stats/k-means-clustering-12.html`