Lab 6 Segmentación

Nathan Mateo Marín
Universidad Los Andes
nm.marin10@uiandes.edu.co

Abstract

En este laboratorio se desarrolla una función de python que segmenta imágenes usando uno de 4 posibles métodos de clustering (k-means, gmm, watershed y jerarquía aglomerativa), los canales que la función usa para segmentar también se le debe proveer colocando como parámetros rgb,lab,hsv,rgb+xy,lab+xy,hsv+xy. Los canales con +xy tienen en cuenta la posición de los píxeles a la hora de segmentar, por eso suelen ocasionar que objetos grandes sean partidos por la mitad. Se llega a la conclusión de que el tipo de algoritmo y el canal a utilizar óptimo dependerá de la imagen a segmentar.

1. Introducción

Lo primero es mencionar que el código y todos sus resultados se encuentran en https://github.com/mateomarin97/IBIO4680/tree/master/06-Segmentation/Answer. Ahí encontrará una carpeta llamada Evaluación con los índices de Jacard calculados para la imagen 24063 usando como anotación la anotación más mínima del set provisto y como imagen segmentada la más mínima generada por cada algoritmo. También hay una carpeta llamada resultados en la cual se encuentran las segmentaciones de las imágenes 24063 y 55067, para cada una de las parejas algoritmo-canal se calcularon 7 segmentaciones (con n clusters 2,3,4,5,6,7,8,9).

Por último el script con la función de segmentación se llama segmentacion.py.

2. Métodos

Empecemos explicando como segmentar una imagen con procesos de clustering. Lo que se hace es que cada píxel se expresa en algún espacio elegido (por ejemplo RGB más la intensidad) luego se decide particionar ese espacio en k grupos que minimicen el desorden (esto se hace con k-means,gmm,watershed o la jerarquía aglomerativa). Finalmente los píxeles pertenecientes a la misma partición se dicen que son parte de un mismo objeto y así se segmenta la imagen en k objetos.

Veamos los algoritmos de clustering.

2.1. K-means

Se elige un número k de puntos al azar (centroides) en el espacio de representación , luego se asigna cada uno de los píxeles al centroide más cercano, esto forma una primera partición, una vez aquí se calcula con los datos de los píxeles el nuevo centroide de cada partición. Estos nuevos centroides se usan para reasignar los píxeles y crear otra partición. El proceso se itera hasta que el algoritmo converja, es decir, de una paso al siguiente no se reasigne ningún píxel a otra partición.

2.2. GMM

Se generan k Gaussianas con media y varianza aleatorias, luego a cada píxel se le asigna una probabilidad de pertenecer a una Gaussiana (viene dada por el valor de dicha Gaussiana en el punto del píxel), luego con estas dependencias especificadas se calcula un nuevo grupo de Gaussianas que se ajusten a ellas. El proceso se repite hasta que converja. Finalmente se dice que un píxel pertenece a la Gaussiana que tenga un mayor valor (probabilidad) en el punto de dicho píxel.

2.3. Watershed

A este método se le deben proporcionar uno marcadores iniciales, en el caso de la función desarrollada dichos marcadores son los mínimos locales de la intensidad de la imagen. El algoritmo se puede entender al visualizar la imagen como una superficie topográfica, donde a cada píxel se le asigna una altura dada por su intensidad (la verdad podría ser cualquier característica del píxel o combinación de características), luego en los puntos designados por los marcadores se empieza a inundar la superficie, cada punto con una tinción de agua distinta, el nivel de los lagos se va subiendo paulatinamente y cuando dos lagos se toquen se dibuja una linea que los separa y se sigue subiendo el agua hasta inundar toda la imagen. Las rayas marcadas se llaman líneas divisoras de aguas y dan los bordes de las regiones de la segmentación, mientras que cada lago distinto es una

región diferente.

2.4. Jerarquía aglomerativa

Se empieza con cada píxel como un grupo diferente, luego se mide la distancia (en el espacio de representación) entre cada dos grupos distintos y se unen los dos grupos más cercanos, se vuelve a medir las distancia entre cada dos grupos distintos y se fusionan de nuevo los dos grupos más cercanos, se sigue iterando hasta que todo los píxeles estén en el mismo grupo. Cada uno de estos pasos se guarda y así luego se puede elegir entre ellos el que tenga el número de grupos deseados.

2.5. Evaluación

La evaluación de este método debe hacerse con cada imagen en particular que se desee segmentar, pues algoritmos y espacios de representación que segmentan bien un tipo de imagen puede que no segmenten tan bien una imagen distinta. En este informe se presenta sólo la evaluación de la función a la hora de segmentar la imagen 24063. El método de evaluación consiste n tomar de cada algoritmos y espacio de representación la segmentación minimal que contenga el objeto de interés (en este caso la capilla) y obtener el índice de Jacard de dicho objeto al compararlo con la forma en que es representado el mismo objeto en la anotación más minimal disponible (que en esta ocasión resulta ser la del sujeto 2).

2.6. Observaciones

Se debe mencionar que el algoritmo de watershed no es capaz de segmentar la imagen en un número especificado de regiones k, sino que la segmenta en superpíxeles, uno por cada mínimo local de la imagen. Sería posible truncar el número de mínimos locales a considerar pero la verdad es un procedimiento que no aporta mucho al algoritmo. Una mejor idea sería dejar que los lagos se unan si se encuentran por debajo de una altura umbral determinada, sin embargo con el algoritmo provisto por Cv2 no es posible hacer esto.

Otra observación es que para el algoritmo de Jerarquía aglomerativa fue necesario recortar las imágenes a un tamaño de 300 X 300, esto se debe a que este algoritmo guarda cada uno de sus pasos, así que mientras los demás guardan una sola segmentación este guarda todas desde 300 X 300 regiones hasta una región. En consecuencia al intentar trabajar con toda la imagen la memoria del computador se acabó.

La última observación compete a la calibración de los espacios de representación. Al usar las funciones de python para pasar de RGB a LAB y HSV se tien que los nuevos canales no van de 0 a 255 (como lo hacen R,G,B) entonces en cada caso se le aplica al canal la transformada requerida para expresarlo de 0 a 255. Por ejemplo: L va de 0 a 100, entonces a todos los valores de L se les multiplica por 255100, A y B van de -128 a 127 así que se les suma 128,etc. Una vez los canales están todos en el mismo rango se puede aumentar o disminuir la influencia de un canal a la hora de segmentar, el truco es que la segmentación se da siempre en basa a un concepto de proximidad entre puntos, entonces si quiero apagar la influencia de un canal lo multiplico por un número grande así las distancias en este canal son más grandes que en el resto y es menos probable que se le tenga en cuenta a la hora de segmentar. Por otro lado, si quiero aumentar la influencia de una canal (como las posiciones de los pixeles en todos los espacios de representación que las tienen en cuenta) lo multiplico por un número entre cero y uno para acortar las distancias en este canal.

2.7. Resultados

Empecemos mostrando la imagen 24063 original y unas cuantas segmentaciones obtenidas de ella

Figure 1. Imagen de la capilla a segmentar, la idea es segmentarla en capilla y cielo, o si se quiere capilla, escalera y cielo

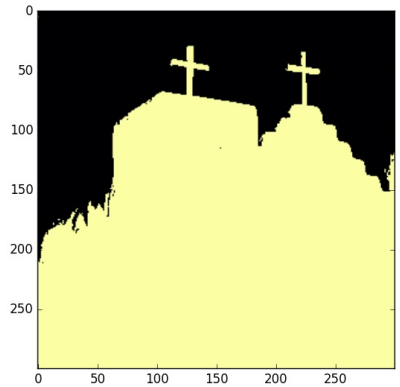

Figure 2. Segmentación obtenida con el método jerárquico y el canal rgb. Indice de Jacard de 96.7013%

Hay varias cosas que observar. Primero se tiene una segmentación adecuada de la capilla tanto en el canal rgb como

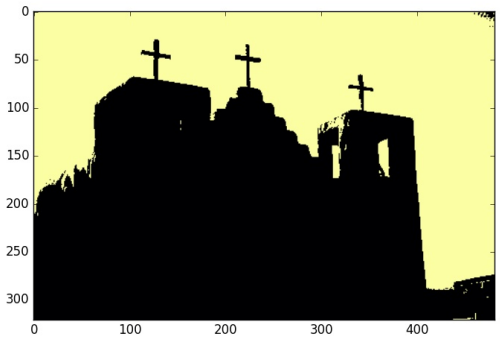

Figure 3. Segmentación obtenida con el método kmeans y el canal lab. Indice de Jacard de 93.283%

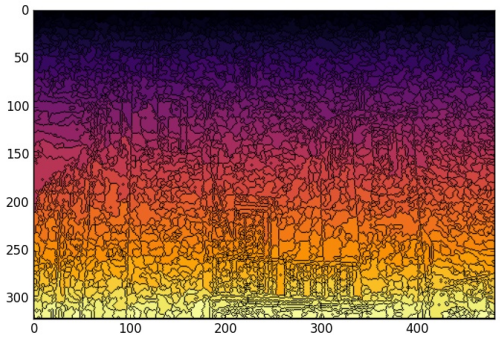

Figure 4. Segmentación obtenida con el método watershed y el canal L como la altura topográfica.

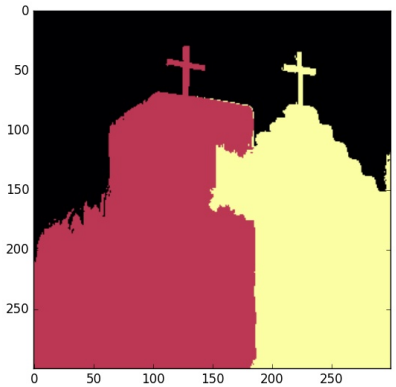

Figure 5. Segmentación obtenida con el método jerárquico y el canal Lab más las posiciones xy.

el lab, sí existe una diferencia entre los dos pero no es muy grande. Segundo el método de watershed en efecto genera una partición de la imagen en superpixeles, lo bueno que tiene es que los bordes de estos superpíxeles se ajustan muy bien a los bordes de los objetos en la imagen. Tercero, cuando se tienen en cuenta las posiciones x,y de los píxeles el algoritmo empieza a dividir objetos grandes en regiones más pequeñas, esto es de esperarse pues intenta minimizar la distancia total entre todos dos puntos de un mismo objeto. Por esta última razón tener en cuenta los canales x,y no es muy útil en el contexto de esta imagen ni de imágenes que tengan regiones muy grandes que representan el mismo objeto.

Se podría pensar que el mismo método de segmentación debe funcionar bien para cualquier imagen, pero tal no es el caso. Las siguientes dos imágenes son un ejemplo

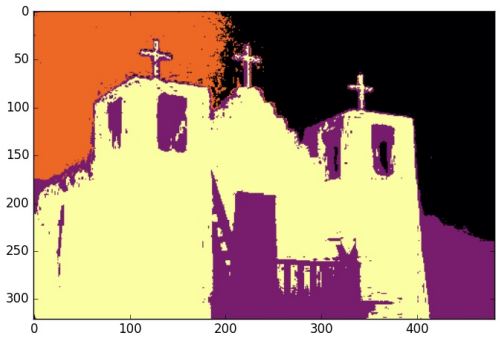

Figure 6. Segmentación obtenida para la capilla con el método GMM y el canal Lab más las posiciones xy.

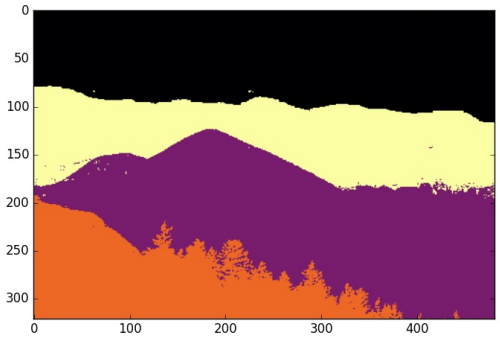

Figure 7. Segmentación obtenida para la montaña con el método GMM y el canal Lab más las posiciones xy.

Claramente en FIG 6 el método GMM de 4 regiones con canales LAB más x,y no funciona bien, pero en FIG 7. Para que quede más claro la imagen que se segmento en FIG 7 fue FIG 8.

Figure 8. Imagen de montañas.

Este ejemplo también sirve para mostrar que el número de regiones en que se quiere segmentar la imagen depende de la imagen que se esta tratando, en especifico, del número de zonas de interés distintas que uno quiera reconocer en la imagen.

3. Conclusiones

La función de segmentación funciona relativamente bien al tener en cuenta lo básicos que son los métodos que implementa. Al conseguir el canal y método óptimo para la imagen de la capilla se llegó a obtener un índice de Jacard de aproximadamente 97% lo cual es bueno. La única mejora que de momento se me ocurre que se pueda hacer es encontrar una forma de truncar el método de watershed, pero hay que ver con mucho cuidado la función de cv2 que hace watershed para determinar como hacerlo.

References