Evaluación de la segmentación de imágenes

Surley Berrio, Victor Cifuentes, Ivan Calvache



Universidad Internacional de la Rioja, Logroño (España)

26 de febrero del 2022

Palabras Clave

Procesamiento de Imágenes, Algoritmos, Otsu, Watershed, Segmentación.

Resumen

El documento se centra en determinar técnicas o algoritmos de segmentación de imágenes para poder identificar fisuras en su asfalto. Se experimenta con un conjunto de imágenes con problemas en su asfalto.

Una vez revisado y clasificado las imágenes, se procederá a la aplicación de los algoritmos de segmentación los cuales son Watershed, Otsu y Adaptive Thresholding. Posterior, se evaluará los rendimientos de estos algoritmos mediante métricas como el indice de Jaccard o la Matriz de Confusión.

Al finalizar el mismo, se pudo observar que el algoritmo de Otsu es el que presenta mejor desempeño. Siendo visibles las zonas afectas en los resultados obtenidos de las imágenes.

I. Introducción

El procesamiento de imágenes se basa en grupo de algortimos para modificar la imagen, esto permite mejorar la calidad para su posterior la toma de decisiones. Entre sus aplicaciones se encuentran en las áreas de la medicina, tecnologia, robotica, entre otras (Gonzalez & Woods, 2008). Estas técnicas de segmentación permiten realizar el procesamiento de imagenes para lo cual es necesario tener el conocimento de la teoría de cada uno de estos.

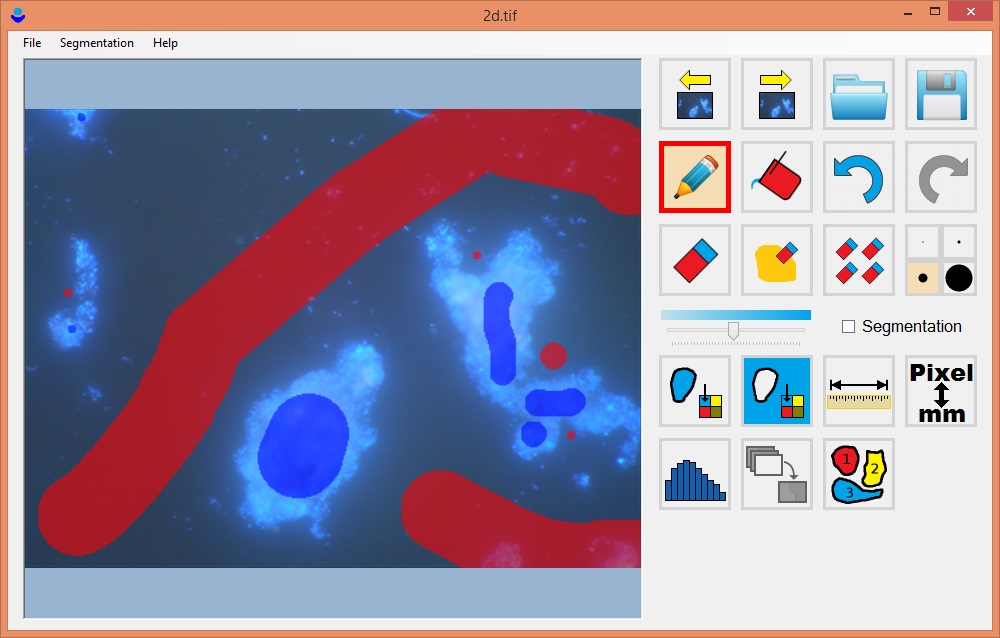
En la sección de la metodología se explicaron los conceptos teóricos de cada segmentador enunciado en este trabajo los cuales son los siguientes:

* Umbral Adaptativo
* Otsu
* Watershed.

En la sección de desarrollo, se aplicó los algoritmos a las imágenes con anomalias. Para aplicar estos algoritmos es necesario contar con imágenes del tipo “ground truth”, a estas también se les suele llamar “Etiquetado Manual de Imágenes” o “Segmentación Manual de Imágenes”, con estas se puede estimar las métricas, para lo cual usaremos una aplicación para la estimación de las mismas;

El aplicativo es Sefexa Image Segmentation Tool la cual es una herramienta libre que permite generar segmentaciones de la imagen, análisis de la imagen y creación del Ground Truth.

Figura 1 Sefexa Image Segmentation Tools



Se hará uso de la aplicación para evaluar al conjunto de imágenes. Las métricas son: el F1-score, Jaccard, RMSE y Matriz de Confusión en los datos binarios de cada pixel de la imagen, es quiere decir, que la imagen tendrá un vector de valores comprendidos entre (0, 1), una de las imágenes esperada, y la otra, de la que resultó al aplicarle alguna de los algoritmo mencionadas de segmentación.

II. metodología

Segmentar una imagen permite separar la imagen en múltiples regiones. Las regiones obtenidas están compuestas de pixeles que tienen características similares, las mismas pueden ser la intensidad, texturas, etc. (Chityala, Ravishankar, 2021). En el mundo del procesamiento de imágenes digitales se puede encontrar con un extenso conjuntos de técnicas de segmentación, las mismas están clasificadas de la siguiente manera:

* Las que se basan en Histogramas
* Las que se basan en Regiones
* Las que se basan en Contornos
* Las que se basan en Borde
* Las que se basan en Clustering

En el siguiente documento se van a evaluar los segmentos basados en Regiones y Contornos con los algoritmos de Otsu, Umbral Adaptativo y Watershed.

1. ***Algoritmo de segmentación: Otsu***

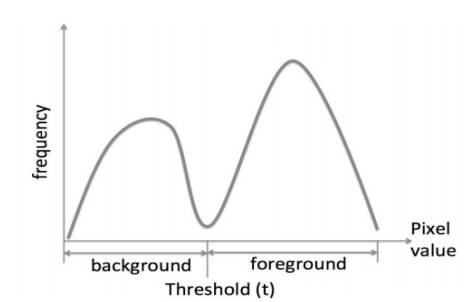
El algoritmo de segmentación Otsu, es uno de los más sencillos, dado su imagen binaria entre [0,255] las que trata de separar clases o grupos de objetos de la imagen y su fondo (Gonzalez, Woods 2018). Siendo L su intensidades, para la imagen de 8 bits, es . Siendo t su valor de umbralización, y su probabilidad para la intensidad calculada, por lo tanto la probabilidad que el píxel este ubicado en el fondo de la imagen es , así también se puede calcular la probabilidad de que el pixel se encuentre en la parte frontal viene dada por donde y serían los promedios de intensidades en el fondo y el frente de la imagen. Se puede decir que , , , es la varianza del grupo de pixeles y se representa por (a) y (b), donde (a) es la varianza para el conjunto del fondo y (b) el conjunto dentro de la imagen las cuales se representan con las siguientes ecuaciones:

(a)

(b)

Este algoritmo indica el umbral óptimo para segmentar una imagen binaria, la varianza es considerada tanto para la representación del fondo ("background") como la del objeto ("foreground"). Ademas, calculamos la varianza de todos los posibles conjuntos y se toma el umbral el cual presenta la máxima varianza entre las clases. Ver figura 2

**Figura 2** Umbral que divide los pixeles



(Chityala, Ravishankar, 2021)

1. ***Algoritmo de segmentación: Adaptive Thresholding***

El valor del umbral global, algunas veces no da una segmentación adecuada o precisa. Este tipo de algoritmo ayuda a que los umbrales adaptativos resuelvan dicho problema. En el umbral adaptativo, la imagen expuesta es dividida en muchas sub-imagenes. (Chityala, Ravishankar, 2021)

El valor del umbral de cada sub-imagen es calculado, esto se utiliza para realizar la segmentación de la siguiente sub-imagen. El nuevo umbral para cada una de las siguientes sub-imagenes sería la media o mediana dependiendo la que se quiera aplicar. Se puede aplicar también algoritmos o procesos personalizados para poder encontrar un valor del umbral adecuado: por ejemplo, el promedio del pixel con valor máximo y mínimo en la sub-imagen (Chityala, Ravishankar, 2021).

1. ***Algoritmo de Watershed***

El algoritmo para la segmentación de Watershed, a la imagen será considerada como un paisaje topográfico (crestas y valles). En paisajes de elevación los valores se definen por los valores grises de sus píxeles. (Preim, Botha 2014)

Para la representación en 3D, el algoritmo de Watershed divide la imagen original en regiones. De cada mínimo local, se tiene una trayectoria más pronunciada para ese mínimo local. Las "regiones" son separadas entre sí. La transformación de regiones descompone la imagen al completo y asigna a cada píxel a una determinada región.

III. DESARROLLO

En esta sección se utilizaron imágenes con anomalías en su asfalto para las respectivas pruebas:

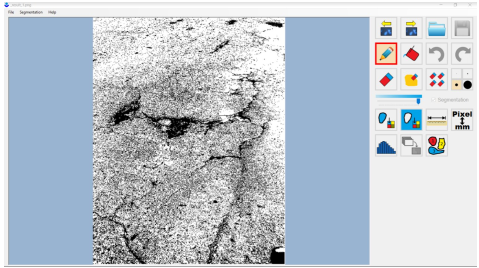
En esta sección se va hacer uso de los conocimientos anteriores para la segmentación se zonas, como se puede observar (ver Figura 3) Este tipo de imagen es considerada de tipo “ground truth” que permita hacer las comparativas y su obtención de sus métricas para evaluar el rendimiento de la misma.



**Figura 3** Fisura en el pavimento de una calle. (Imagen Original)

1. ***Obtención del Ground Truth de la Imagen***

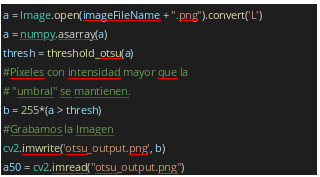
Se usará la herramienta *Sefexa Image Segmentation Tool* la que en su conjunto de herramientas tiene un segmentador en tiempo real, para poder modificar sus parámetros y así poder visualizar su resultado. La imagen resultante se puede observar en la figura 4.



**Figura 4** Uso del Aplicativo para la obtención del Ground Truth

1. ***Algoritmo de Segmentación: Otsu***

Se hará uso de la libreria en Python skimage ya que tiene implementado una función para realizar la segmentación Otsu (Figura 5)



**Figura 5** Segmentación de Otsu: implementación con Python y SKImage

Se procede a realizar la prueba con la imagen de la derecha de la figura 6 y su respectiva imagen resultante se tiene al lado derecho. Adicional, para procesar es necesario realizar la binarización de la imagen.



**Figura 7** Segmentación de Otsu: Ejecución con Python y SKImage.

El algoritmo de Otsu, usa el histograma de la imagen de entrada para calcular su umbral.

1. ***Metodología de Segmentación Umbral Adaptativo (Adaptive Thresholding)***

Para implementar en Python la segmentación Umbral Adaptativo, se empleó una de las funciones de la librería OpenCv. Ver Figura 6

#Abrimos la imagen  y la convertimos a escala de grises

a = Image.open(imageFileName + '.png').convert('L')

a = numpy.asarray(a)  #La imagen se lee y se establece un umbral

b = cv2.adaptiveThreshold(a,a.max(), cv2.ADAPTIVE\_THRESH\_MEAN\_C, cv2.THRESH\_BINARY,11,1)

cv2.imwrite(imageFileName + '\_adaptive\_output.png', b)

#Cargamos las tres imagenes Original, Segmentada , y la esperada

a40 = cv2.imread(imageFileName + "\_adaptive\_output.png")

Figura 6 Implementación en Python de la segmentación por Umbral Adaptativo

En la Figura 7 se observa la imagen de entrada, que dentro de la implementación se binariza para poder ser procesada.

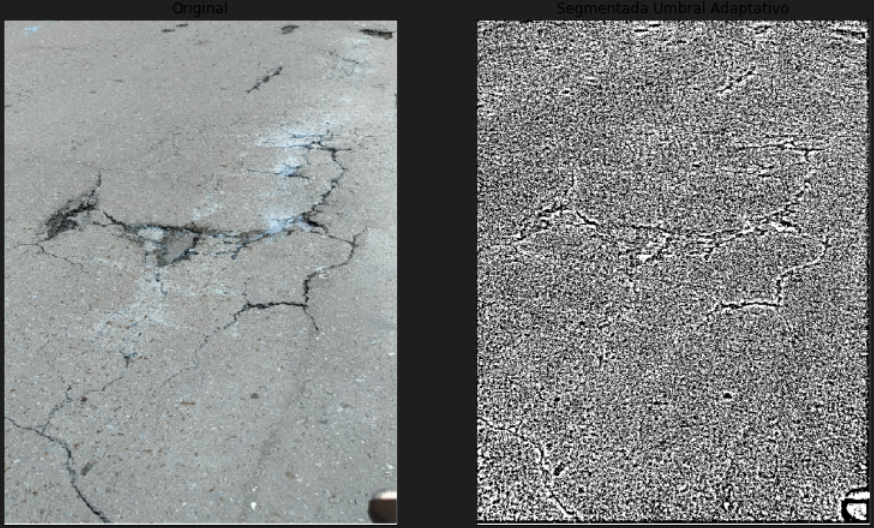


Figura 7 Implementación en Python de la segmentación por Umbral Adaptativo.

Para esta aplicación se utilizó el método adaptativo de la media.

1. ***Metodología de Segmentación Watershed***

Para implementar en Python la segmentación Watershed, se uso una de las funciones de la librería OpenCv. Ver Figura 8.

Previo al procesamiento de la imagen directamente con la función cv de watershed, es necesario realizar un filtrado previo, para este caso se usó:

img\_original = cv2.imread(imageFileName + '.png')

gray = cv2.cvtColor(img\_original,cv2.COLOR\_BGR2GRAY)

thresh, b1 = cv2.threshold(gray, 0, 255,  cv2.THRESH\_BINARY\_INV+cv2.THRESH\_OTSU)

b2 = cv2.erode(b1, None,iterations = 2)

dist\_trans = cv2.distanceTransform(b2, 2, 3)

thresh, dt = cv2.threshold(dist\_trans, 1,  255, cv2.THRESH\_BINARY)

labelled, ncc = label(dt)

cv2.watershed(img\_original, labelled)

#guardar imagen segmentada watershed

cv2.imwrite('img\_labelled.png', labelled)

#lectura imagen watershed

segm\_water = cv2.imread('img\_labelled.png')

#binarización

ret,thresh1 = cv2.threshold(segm\_water,40,255,cv2.THRESH\_BINARY)

Figura 8 Implementación en Python de la segmentación por Método Watershed

En las Figura 9 se observa la aplicación del Método Watershed para segmentación de la imagen de prueba



Figura 9 Implementación en Python de la segmentación por Waterhed.

1. ***Cálculo de las Métricas (RMSE, F1-Score, Jaccard y Matriz de Confusión)***

Para el cálculo de las métricas, se consideró la revisión píxel a píxel, en este caso, se tomó la imagen segmentada la cual consideramos como la “Predicción”, y la imagen de ground truth, que serían los valores esperados. De manera practica se procedió a convertir la matriz binaria a una sola lista de valores para que puedan realizar los cálculos correspondientes (3)

(3)

***RMSE***.

Considere n como el total de pixeles en al arreglo transformado y sea el valor del píxel en la posición i de la imagen segmentada, y el valor del píxel en la posición i de la imagen de groundtruth. Por tanto el MSE o (Error cuadrático medio) viene dado por la ecuación 4 y el RMSE por la ecuación 5:

(4)

Cuando el MSE es 0 se considera que las imágenes son idénticas, con el RMSE Esta métrica amplifica y penaliza los errores grandes. En Python usando la librería sklearn.metrics tenemos la función que describe la figura 10 :

mse = mean\_squared\_error(groundTruth, resp, squared=False)

rmse = math.sqrt(mse)

Figura 10 Funciones Python para el cálculo de RMSE

Esta función recibe las dos imágenes que han sido transformad en un solo arreglo unidimensional. Luego obtenemos la raíz cuadrada.

***F1-SCORE***

La puntuación del F1 toma valores entre 0 y 1 y se calcula como lo describe la ecuación 5:

(6)

Siendo 0 el peor y 1 la mejor predicción, esto como base de una matriz de confusión de las dos imágenes tanto de la segmentada como la de groundtruth (Ver Figura 11)

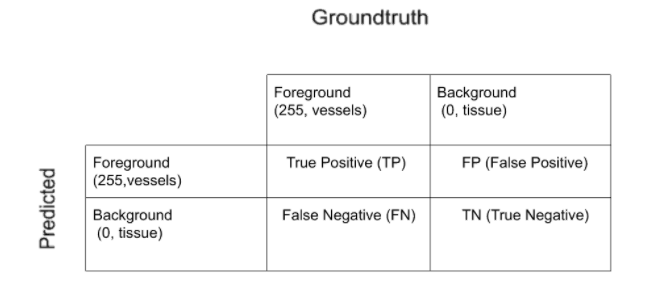


Figura 11 Matriz de confusión para comparar la imagen segmentada vs la de Groundtruth ([Vemuri](https://www.linkedin.com/in/pranathi-vemuri-ms/" \t "_blank), 2019)

La implementación en Python para el calculo del Fscore usamos la función de la librería sklearn.metrics. (Ver figura 11)

f1 =  f1\_score(groundTruth, resp, average='macro')

Figura 11 Funciones Python para el cálculo de F1-score

Así mismo el F1-score recibe las dos imágenes, la groundtruth y la segmentada, adicional de un parámetro del balanceo de los pesos de las clases de la imagen, en este caso 0 y 1

***MATRIZ DE CONFUSION***

En el campo de la inteligencia artificial y en especial en el problema de la clasificación estadística, una matriz de confusión es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo dos clases (Wikipedia, 2019)

IV. ANÁLISIS DE RESULTADOS

En cuanto al rendimiento de los segmentadores propuestos en este trabajo, se muestran los resultados en las tablas 1 y 2

Tabla 1 Métricas del rendimiento de los segmentadores

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***Técnicas de Segmentación*** | | |
| **Otsu** | **Umbral Adaptativo** | **Watershed** |
| **RMSE** | 0.41 | 0.69 | 0.72 |
| **F1-Score** | 0.96 | 0.75 | 0.47 |
| **Jaccard** | 0.93 | 0.60 | 0.38 |
| **Accuracy** | 0.97 | 0.77 | 0.72 |

Con estos valores de métricas podemos ver claramente el segmentador que resultó con mejor rendimiento para estas imágenes, fue el método Otsu que llegó a segmentar de manera aproximada a su imagen de ground truth.

Situándose por en medio de las dos técnicas se encuentra el de Umbral Adaptativo, si bien es cierto su aproximación podría mejorar en este ejercicio manipulando los parámetros, no se logró una buena segmentación.

El peor puntuado, de acuerdo con las métricas de rendimiento fue el método de Watershed, sin embargo, esto no quiere decir que el método en sí sea malo, este método está más orientado para realizar segmentaciones sobre imágenes con texturas homogéneas, como por ejemplo en imagenología médica.

Para aplicar el criterio de desempate entre los segmentadores para elegir el valor de los píxeles, lo que se realizo fue trabajar posición por posición y sumar sus valores. Los que fueran mayores a la media proceden a pertenecer al rango más alto de la escala (255) caso contrario pertenecen al rango menor (0). De esta manera se pudo obtener un nuevo arreglo con los píxeles elegidos determinado por el criterio.

En la tabla 2 se pueden observar los resultados de la integración por votación de los 3 métodos

Tabla 2 Métricas del rendimiento de segmentador por votación

|  |  |
| --- | --- |
|  | ***Segmentación por Votación*** |
| **F1-Score** | 0.95 |
| **Jaccard** | 0.90 |
| **Accuracy** | 0.96 |

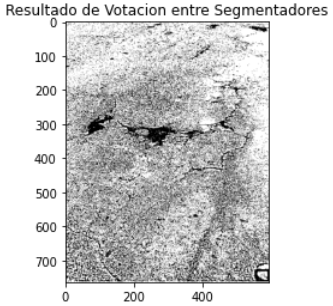


Figura 12 Resultado de votación entre segmentadores

V. CONCLUSIONES

* El trabajo evidenció que para estos segmentadores, en este tipo de imágenes se destaca el método de Otsu
* Al contar con imágenes de groundtruth y es posible realizar comparaciones del rendimiento de los diferentes algoritmos a través de métricas como el RMSE, f1-score, Jaccard y Matriz de Confusión.
* El método de Watershed no representa un buen segmentador ante este tipo de imágenes. Sin embargo, es muy útil en áreas de imagenología médica.
* Luego de evaluar los segmentadores se implementó una clasificación por votación donde se obtuvo un resultado similar al algoritmo de Otsu.

.

Referencias

Gonzalez, R. C., & Woods, R. E. (2008). Digital Image Processing Second Edition Education 2012 MSU LIBRARY.

Chityala, Ravishankar (2021). Image Processing and Acquisition using Python (Chapman & Hall/CRC The Python Series) (p. 167-180). CRC Press. Edición de Kindle.

R.C. Gonzalez, R.E. Woods (2018), S.L. Eddins, in Digital Image Processing (Pearson Prentice Hall, Upper Saddle River, NJ,), chap. 10

Preim B., Botha C. (2014), in [Visual Computing for Medicine (Second Edition)](https://www.sciencedirect.com/book/9780124158733/visual-computing-for-medicine), chap 9-10

Vremuri, P. (2019, July 18). Image Segmentation with Python. Kite. <https://www.kite.com/blog/python/image-segmentation-tutorial/>

Wikipedia. (2019). Matriz de confusión - Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/Matriz_de_confusión>

Sefexa (1.2.2.3). (2020). [Semi-automatic image segmentation]. Ales Fexa. http://www.fexovi.com/ales.html