Evaluación de distintos tipos de algoritmos orientados a la segmentación de imágenes

Miguel Murillo, Iván Calvache, Diego Tamayo, Víctor Cifuentes

Universidad Internacional de la Rioja, Logroño (España)

19 de febrero del 2023

RESUMEN

El presente artículo se centra en determinar técnicas o algoritmos de segmentación de imágenes para poder identificar fisuras en asfalto. Se experimenta con un conjunto de imágenes con problemas en su asfalto. Una vez revisadas y clasificadas las imágenes, se procederá a la aplicación de los algoritmos de segmentación los cuales son Watershed, Otsu y Adaptive Thresholding. Posteriormente, se evaluarán los rendimientos de estos algoritmos mediante métricas como el índice 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.

PALABRAS CLAVE

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

I. INTRODUCCIÓN

El procesamiento de imágenes se basa en grupos de algoritmos 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, tecnología, robótica, entre otras (Gonzalez & Woods, 2008). Estas técnicas de segmentación permiten realizar el procesamiento de imágenes, para lo cual es necesario tener el conocimiento 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 anomalías. 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.

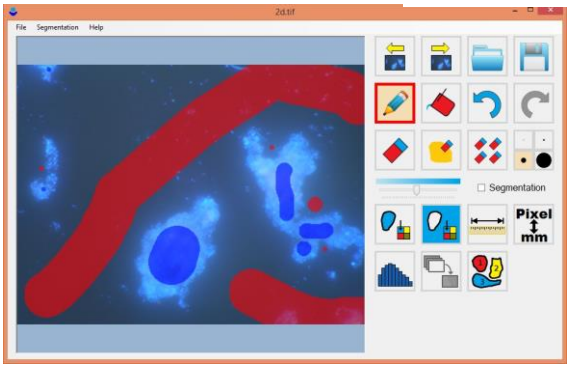

Fig. 1 Sefexa Image Segmentation Tools

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 algoritmos 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 conjunto 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 presente trabajo se van a evaluar los segmentos basados en Regiones y Contornos con los algoritmos de Otsu, Adaptive Threshold y Watershed.

A. Algoritmo de segmentación: Otsu

El algoritmo de segmentación Otsu, es uno de los más sencillos, dada una imagen binaria entre [0,255] trata de separar clases o grupos de objetos de la imagen y su fondo (Gonzalez, Woods 2018). Siendo L sus intensidades, para la imagen de 8 bits, es $L = 2^8 = 256$. Siendo t su valor de umbralización, y p_i su probabilidad para la intensidad calculada, por lo tanto, la probabilidad que el píxel este ubicado en el fondo de la imagen es $P_b(t) = \sum_{i=0}^{t} p_i$, así también se puede calcular la probabilidad de que el pixel se encuentre en la parte frontal viene dada por $P_f(t) = \sum_{i=t+1}^{L-1} p_i$ donde m_b y m_f serían los promedios de intensidades en el fondo y el frente de la imagen. Se puede decir que v_b, v_{fb}, v, 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:

$$v_{dentro} = P_b(t)v_b + P_{bf}(t)v_f \quad \text{(a)}$$

$$v_{entre} = v - v_{dentro} \quad \text{(b)}$$

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"). Además, calculamos la varianza de todos los posibles conjuntos y se toma el umbral el cual presenta la máxima varianza entre las clases. Ver fig. 2

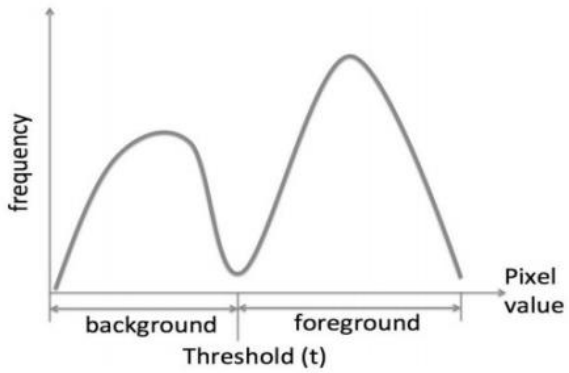

Fig. 2 Umbral que divide los pixeles

B. Algoritmo basado en la segmentación de umbral de tipo adaptativo (Adaptive Threshold)

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).

C. 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 fig. 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.

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

A. 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 fig. 4.

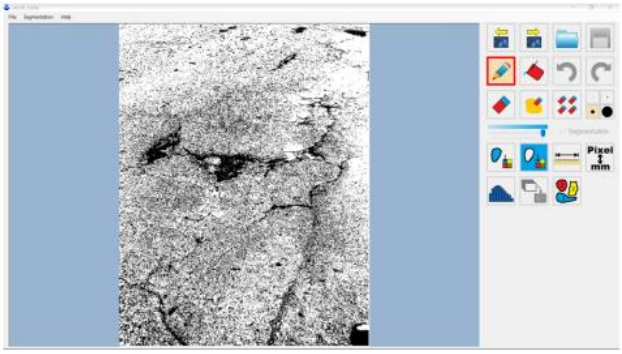

Fig. 4 Uso del Aplicativo para la obtención del Ground Truth

B. Algoritmo de Segmentación: Otsu

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

```
a = Image.open(imageFileName + ".png").convert('L')
a = numpy.asarray(a)
thresh = threshold_otsu(a)
#Píxeles con intensidad mayor que la
# "umbral" se mantienen.
b = 255*(a > thresh)
#Grabamos la Imagen
cv2.imwrite('otsu_output.png', b)
a50 = cv2.imread("otsu_output.png")
```

Fig. 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.

Fig. 6 Segmentación de Otsu: Ejecución con Python y SKImage.

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

C. Metodología basada en la segmentación de umbral de tipo adaptativo (Adaptive Threshold)

Con el objetivo de poder implementar la metodología basada en la segmentación de umbral de tipo adaptativo a través del lenguaje de programación Python se recurrió a una de las implementaciones que se encuentran en una de las librerías más famosas, OpenCV. Esto se lo puede observar en fig. 6.

```
#Se abre la imagen y se la transforma 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 imágenes que son: Original,
Segmentada y esperada
a40 = cv2.imread(imageFileName +
"_adaptive_output.png")
```

Fig. 7 Uso de OpenCV para establecer una posible implementación en Python de la metodología basada en la segmentación de umbral de tipo adaptativo (Adaptive Threshold)

Se puede observar a partir de fig. 7 la necesidad de aplicar una operación de binarización a la imagen original ya que es un pre requisito para su posterior procesamiento.

Es necesario afirmar que de entre todas las opciones de métodos de segmentación de umbral de tipo adaptativo, se utilizó aquella basada en el uso de la media.

Fig. 8 Resultado obtenido luego de aplicar la metodología basada en la segmentación de umbral de tipo adaptativo (Adaptive Threshold) a una imagen

D. Metodología basada en la segmentación con el algoritmo de Watershed

El algoritmo Watershed es uno de los más populares y utilizados para la segmentación de imágenes, en el presente

trabajo se realizó una implementación del mismo en el lenguaje de programación Python tomando como punto de partida los beneficios y métodos provistos por la librería OpenCV.

Sin embargo, antes de enviar la imagen original o input a las funciones de Watershed disponibles en la librería, fue necesario realizar un pre procesamiento de la misma.

```
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 la imagen segmentada por Watershed
cv2.imwrite('img_labelled.png', labelled)
#Lectura de la imagen Watershed
segm_water = cv2.imread('img_labelled.png')
#b¿Binarización
ret,thresh1 =
cv2.threshold(segm_water,40,255,cv2.THRESH_BINARY)
```

Fig. 9 Uso de OpenCV para establecer una posible implementación en Python de la metodología basada en el algoritmo Watershed

A continuación, se ilustran los resultados de la aplicación del algoritmo Watershed a una imagen arbitraria.

Fig. 10 Resultado obtenido luego de aplicar la metodología basada el algoritmo Watershed a una imagen

E. Estimación de las métricas de evaluación más adecuadas

Entre las métricas de evaluación escogidas para este estudio se encuentran la raíz cuadrada del error cuadrático medio (RSME), el valor F1 (F1-score), la matriz de Jaccard y la matriz de confusión. En el cómputo de estas métricas de evaluación se realizó una validación en cada uno de lo pixeles de la imagen previamente segmentada. Esta imagen representa aquello denominado como predicción. Por otro lado, se tiene la imagen Ground Truth que representa aquellos valores que se desea o espera obtener. El cálculo se realizó transformando la matriz de valores binarios a un vector unidimensional.

Raíz cuadrada del error cuadrático medio (RMSE)

Si se tiene un determinado número de píxeles en el vector unidimensional y si $P_{segmentado}(i)$ es el valor que posee un determinado pixel en una posición arbitraria i dentro de la imagen que ha sido previamente segmentada. Si $P_{groundtruth}(i)$ representa el valor de un determinado pixel en la misma posición i en la imagen Ground Truth, el error cuadrático medio se define de la siguiente forma:

$$MSE = \sum_{i=1}^{n} (P_{groundtruth}(i) - P_{segmentado}(i))^2$$

Por lo tanto, la raíz cuadrada del error cuadrático medio se define de la siguiente forma:

$$RMSE = \sqrt{MSE}$$

La conclusión que se puede obtener en base a esta métrica se puede definir por el hecho que si $MSE = 0$ las imágenes son iguales, la tarea del RMSE es castigar de cierta forma los errores mayores que se obtuvieran durante el cálculo. Python posee una librería de aprendizaje automático muy popular la cual es Scikit-learn, dentro de ella se puede encontrar el módulo "metrics" que permite calcular el RMSE de un determinado valor de entrada. Esos valores de entrada son las imágenes que previamente han sido pasadas a un vector de una sola dimensión.

```
mse = mean_squared_error(groundTruth, resp,
squared=False)
rmse = math.sqrt(mse)
```

Fig. 11 Cálculo del RMSE a través del módulo "metrics" de Scikit-learn

Valor F1 (F1-score)

Esta métrica consiste en arrojar valores comprendidos entre el 0 y el 1. Su ecuación se ilustra a continuación:

$$F1 = \frac{2 * TP}{2 * TP + FP + FN}$$

Cuando el resultado es 0 la predicción es muy mala, mientras que si es 1 la predicción es excelente. Evidentemente los valores utilizados para este cómputo se obtienen de la matriz de confusión de la imagen segmentada y de la imagen Ground Truth pues es necesario tener los verdaderos positivos (TP), los falsos positivos (FP) y los falsos negativos (FN).

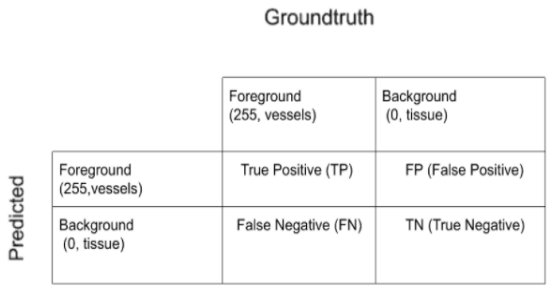

Fig. 12 Matriz de confusión de la comparativa de la imagen segmentada y la imagen Ground Truth

Para poder obtener el valor F1 se utilizó la función correspondiente ya implementada dentro del módulo "metrics" de la librería Scikit-learn.

La función toma como parámetros a ambas imágenes, las cuales son la imagen segmentada y la imagen Ground Truth, además de otro valor que es necesario dentro de la implementación y que oscila entre el valor de 0 y el valor de 1.

```
f1 =  f1_score(groundTruth, resp, average='macro')
```

Fig. 13 Cálculo del valor F1 a través del módulo "metrics" de Scikit-learn

Matriz de confusión

Dentro de aquello que engloba los problemas de clasificación en aprendizaje automático, se tiene el concepto de matriz de confusión, la cual provee una estimación del rendimiento de un determinado modelo basado en un algoritmo de aprendizaje supervisado. Las columnas de esta matriz se relacionan al número de predicciones sobre cada clase, por otro lado, las filas se relacionan al número de clases verdaderas.

IV. RESULTADOS

Luego de realizar la evaluación del experimento, se obtuvieron los resultados de las segmentaciones los cuales se muestran en la tabla 1 y la tabla 2.

	Técnicas de Segmentación		
	Otsu	Adaptive Threshold	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

Tabla 1 Métricas resultantes de los distintos segmentadores utilizados

En base a los valores obtenidos por todas las métricas utilizadas se puede evidenciar cuál de los segmentadores obtuvo los mejores resultados, el mismo fue el método Otsu el cual obtuvo los resultados más similares a los de Ground Truth.

El segundo mejor segmentador fue el Adaptive Threshold, mientras que el peor fue el algoritmo Watershed. A pesar de todas estas consideraciones no se puede afirmar que el algoritmo sea pésimo, sino que más bien está orientado a segmentar imágenes que poseen texturas con tendencia de tipo más homogénea.

Posteriormente a la obtención de los resultados de las métricas, fue necesario buscar una manera de dar una evaluación más, la cual se basó en trabajar con los pixeles de las imágenes segmentadas y sumar sus valores por cada una de sus diferentes ubicaciones o posiciones. Aquellos pixeles con valores superiores a la media se categorizan en el rango más elevado que es 255 y aquellos con valores inferiores a la media se categorizan en el rango más bajo el cual es el valor nulo o cero. A través de este segundo criterio se logró conseguir un vector que contiene cada uno de los pixeles que fueron seleccionados. Los resultados de este segundo criterio se ilustran en la tabla 2.

	Segmentación por Votación
F1-Score	0.95
Jaccard	0.90
Accuracy	0.96

Tabla 2 Métricas resultantes luego del criterio de las votaciones de los segmentadores

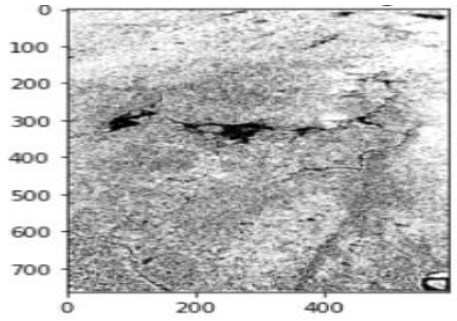

Figura 14 Resultado obtenido luego del criterio de las votaciones de los segmentadores

V. CONCLUSIONES

- El uso de imágenes de tipo Ground Truth provee la posibilidad de efectuar evaluaciones a través de métricas idóneas como lo son la raíz cuadrada del error cuadrático medio (RMSE), el valor F1 (F1-score), la matriz de Jaccard y la matriz de confusión.
- El método de Otsu se presentó como el segmentador más conveniente y adecuado para imágenes de grietas en calles.

- El algoritmo Watershed puede considerarse como el peor al momento de intentar realizar una segmentación de imágenes de grietas en calles.
- Además de realizar la evaluación de la segmentación se decidió realizar un posterior análisis a través del criterio de las votaciones en base a la suma de los valores de los pixeles en las imágenes segmentadas y su posterior categorización.

VI. BIBLIOGRAFÍA

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

[2] Gonzalez R., Woods R. (2018). S.L. Eddins, in Digital Image Processing.

[3] Gonzalez R., Woods R. (2008). Digital Image Processing Second Edition Education 2012 MSU LIBRARY.

[4] Chityala R. (2021). Image Processing and Acquisition using Python. Chapman & Hall/CRC The Python Series.

[5] Narkhede S. (2018). Understanding confusion matrix. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

[6] Vremuri P. (2019). Image Segmentation with Python. https://www.kite.com/blog/python/image-segmentation-tutorial/

[7] Preim B., Botha C. (2014). Visual Computing for Medicine - Second Edition.

[8] Eddins S. (2018). The Watershed Transform: Strategies for Image Segmentation.

[9] Splunk.com (2022). What Is Adaptive Thresholding? https://www.splunk.com/en_us/data-insider/adaptive-thresholding.html