Proyecto intermedio: Fundamentos de Procesamiento Digital de Imágenes

Karla Paola Martínez Velázquez
Eduardo Ramírez de la Fuente
David Pedroza Segoviano

Nota: El código se realizó en Python. Y está hecho de tal manera que se puede leer al mismo tiempo que el reporte, debido a que está hecho en un notebook.

INTRODUCCIÓN

La detección de líneas de una imagen, el cálculo automático de distancias, y el reconocimiento facial de los dispositivos modernos son procedimientos que en algún punto de su procesamiento incluyen un proceso para detectar y segmentar objetos.

Este proyecto se hizo con el objetivo de aprender el uso de estas herramientas de procesamiento de imágenes con algunos de sus métodos y funciones más conocidas y útiles, empezando con una imagen predefinida, se utilizaron dichas funciones para encontrar objetos contenidos en ella, así como la mayor cantidad de propiedades y mediciones que son capaces de dar.

METODOLOGÍA

Procesamiento de imágenes: Es un conjunto de técnicas que se aplican a las imágenes para obtener información mejorada o para extraer alguna información útil. En este estudio, se utilizó el procesamiento de imágenes para convertir la imagen original en una imagen binaria y para normalizar la imagen en blanco y negro.

Segmentación de imágenes: Es el proceso de dividir una imagen digital en múltiples segmentos para simplificar la representación de la imagen en algo que sea más significativo y más fácil de analizar. En este estudio, se segmentaron los triángulos y las aristas de los triángulos en la imagen.

Cálculo de propiedades de regiones: Es el proceso de calcular ciertas propiedades, como el área y el centroide, de las regiones de una imagen. En este estudio, se calculó el área y el centroide de cada triángulo en la imagen.

Transformación de Hough: Es una técnica que se utiliza para detectar formas simples, como líneas y círculos, en una imagen. En este estudio, se utilizó la transformación de Hough para visualizar las líneas de forma individual y para identificar los puntos más representativos de la transformación.

Detección de bordes: Es una técnica de procesamiento de imágenes que se utiliza para identificar los puntos en una imagen donde hay un cambio abrupto en la intensidad de color. En este estudio, se utilizó la detección de bordes para segmentar las aristas de cada triángulo.

Cálculo de áreas y centroides de triángulos: Es el proceso de calcular el área y el centroide de un triángulo a partir de la longitud de sus lados y la ecuación de sus rectas. En este estudio, se utilizó la fórmula de Herón para calcular el área de los triángulos y se calcularon los puntos medios de las rectas para calcular los centroides.

Identificación de líneas paralelas: Es el proceso de identificar líneas que son paralelas entre sí en una imagen. En este estudio, se identificaron grupos de líneas paralelas entre diferentes triángulos.

Uso de pandas para manipulación de datos: Pandas es una biblioteca de Python que proporciona estructuras de datos y funciones de alto nivel para hacer el trabajo con datos estructurados o tabulares más fácil. En este estudio, se utilizó la función "pivot_table" de pandas para agrupar cada arista con su triángulo correspondiente.

Morfología: En matemáticas, la morfología es una herramienta que permite extraer componentes de las imágenes que son útiles para representar y describir formas y bordes, entre otras cosas. Además de las operaciones fundamentales de los operadores morfológicos (erosión y dilatación) existen más técnicas tal como el etiquetado de componentes conectados en una imagen.

Un píxel con las coordenadas (x,y) tiene dos vecinos horizontales y dos vecinos verticales. A este set se le denomina como $N_4(p)$, a los cuatro vecinos diagonales se les llama $N_D(p)$ y a la unión de dichos conjuntos, que son los 8 vecinos de p, se les conoce como $N_8(p)$. La conexión entre componentes depende del tipo de vecinos que son uno del otro (refiriéndose a si son diagonales u horizontales y verticales).

Etiquetado de componentes conectados: La función bwlabel detecta todos los componentes conectados y los almacena en una imagen binaria donde f es la imagen de entrada y conn especifica la conectividad (8 o 4) requerida entre píxeles para determinar si estos están conectados.

Complemento de imágenes: IM2 = imcomplement(IM) procesa el complemento de la imagen dada IM. IM puede ser una imagen de tipo binario o RGB. IM2 es de la misma clase y tamaño que IM.

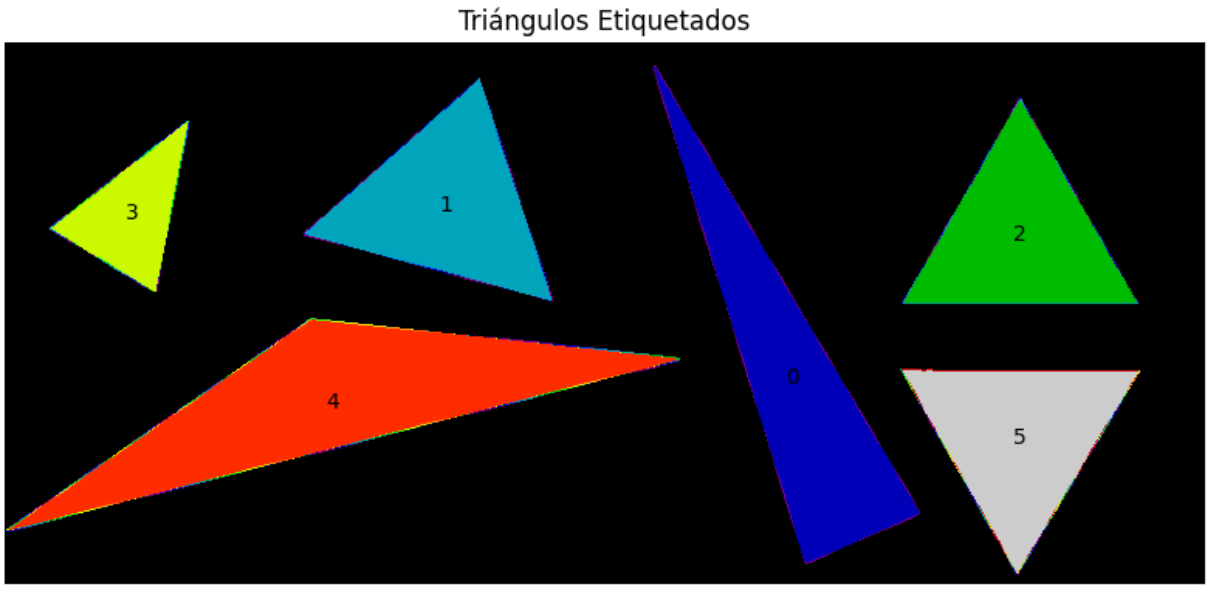
En el complemento de la imagen binaria, los ceros se vuelven unos y viceversa, el negro y blanco son revertidos. En el inverso de una imagen RGB, cada valor de píxel es sustraído del valor máximo de pixel soportado o la clase (o 1.0 para las imágenes de doble precisión) y la diferencia es usada como el píxel en la imagen de salida. En la imagen de salida, las áreas oscuras se vuelven más claras y viceversa.

Si IM es una imagen RGB de clase doble, se puede usar la expresión 1-IM en lugar de esta función. Si IM es una imagen binaria, se puede usar la expresión ~IM en lugar de esta función.

RESULTADOS

1. (2 puntos) Segmentar los triángulos de la imagen, calcule el área y el centroide para cada triángulo. Haga una tabla con esta información

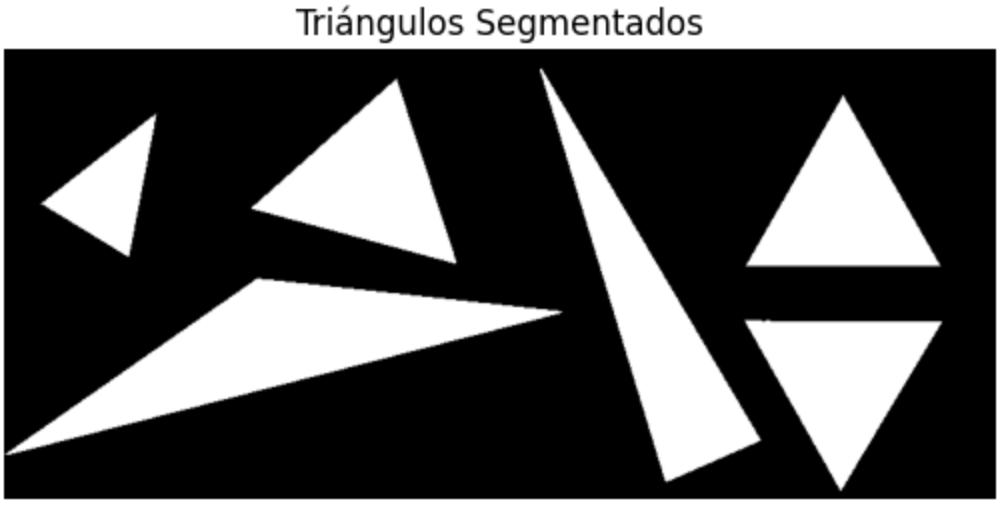
Resultados:



	area	centroid_row	centroid_col
0	15013	225.923600	527.575368
1	11491	108.808024	295.009747
2	10936	129.330468	680.500000
3	4664	114.913593	84.635506
4	20913	242.733515	219.636877
5	11042	266.330647	679.994113

Explicación:

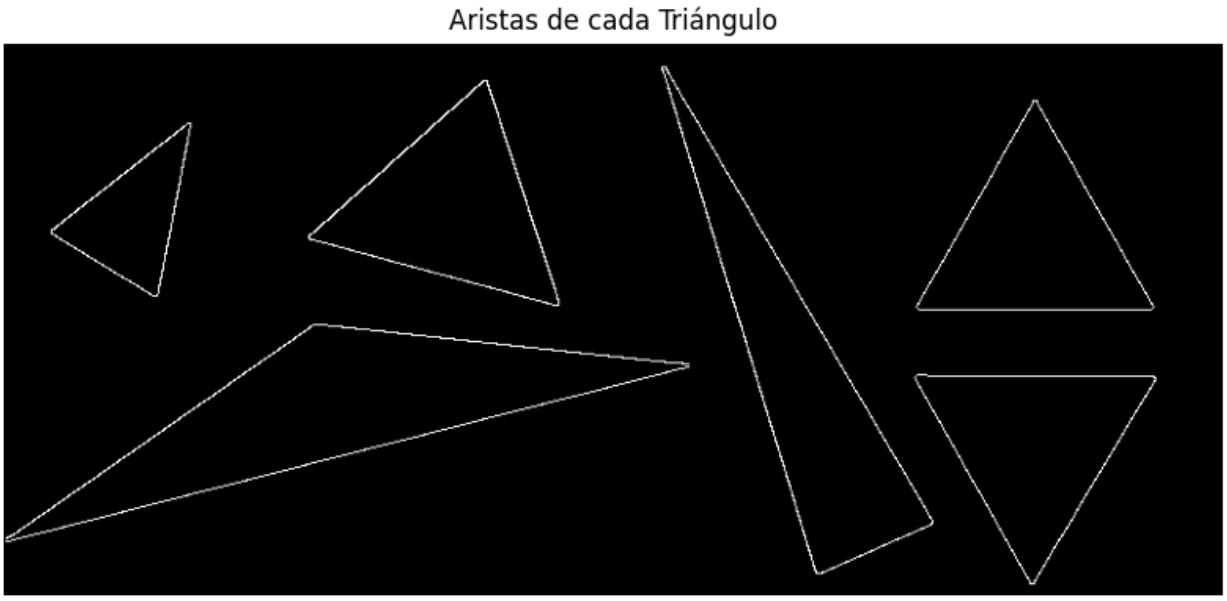
Para este paso, primero se hizo uso de la función imcomplement, para pasar de la imagen original a la imagen binaria siguiente:



A continuación, se aplicó la función **`label()`** de la biblioteca skimage, que es equivalente a la función **`bwlabel()`** de MATLAB. Posteriormente, se empleó la función **`regionprops_table()`** de la misma biblioteca para calcular las propiedades de los objetos en la imagen binaria. Se seleccionaron las propiedades "area" y "centroid". Estas propiedades fueron utilizadas para enumerar cada uno de los triángulos, como se muestra en la figura de los resultados. Esta enumeración facilitó un seguimiento más claro de las funcionalidades posteriores.

2. (2 puntos) Segmentar las aristas de cada triángulo. En los resultados mostrar la imagen donde se pinten las aristas encontradas

Resultados:



Explicación:

Antes de aplicar la función `**canny()**` de la biblioteca skimage, se realizó la normalización de la imagen en blanco y negro. Esta etapa de normalización fue necesaria para preparar la imagen antes de su procesamiento con la función `**canny()**`. Tras aplicar esta función, se obtuvieron los resultados esperados.

3. (2 puntos) Encontrar los modelos de los segmentos de rectas que forman cada arista y para cada triángulo. Los parámetros que debe encontrar de cada arista son: θ, ρ, longitud. Hacer una tabla donde cada renglón sea una arista (identifíquelas con un número) y las columnas sean los parámetros indicados.

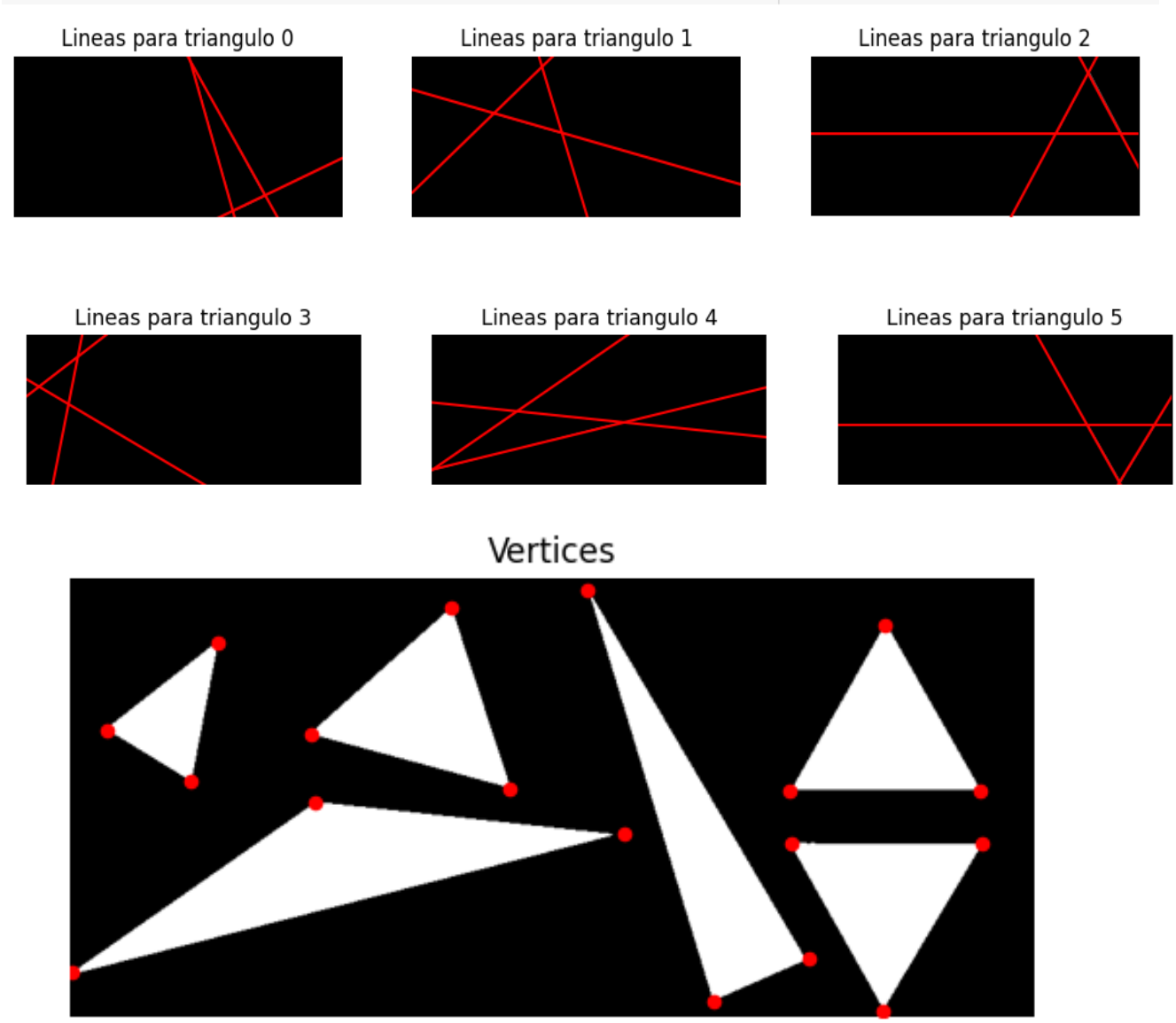


Tabla de resultados:

	Triangle	Theta	Theta (Grados)	Rho	Distances
A1	T0	1.150000	65.890146	540.0	87.085975
A2	T0	-0.296706	-17.000000	410.0	357.375832
A3	T0	-0.541052	-31.000000	365.0	357.207608
A4	T1	-1.308997	-75.000000	-73.0	172.317123
A5	T1	0.837758	48.000000	230.0	158.193878
A6	T1	-0.314159	-18.000000	295.0	158.193878
A7	T2	-1.570796	-90.000000	-176.0	159.348674
A8	T2	0.523599	30.000000	608.0	159.348674
A9	T2	-0.523599	-30.000000	570.0	159.348674
A10	T3	0.191986	11.000000	131.0	117.945106
A11	T3	0.907571	52.000000	117.0	118.717298
A12	T3	-1.029744	-59.000000	-92.0	82.884077
A13	T4	1.326450	76.000000	318.0	474.910863
A14	T4	0.959931	55.000000	269.0	247.583029
A15	T4	-1.466077	-84.000000	-163.0	259.416950
A16	T5	-1.570796	-90.000000	-220.0	160.438500
A17	T5	0.541052	31.000000	766.0	162.030667
A18	T5	-0.506145	-29.000000	419.0	158.797462

Explicación:

En esta etapa, se empleó la transformación de Hough para cada triángulo con el propósito de visualizar las líneas de forma individual, tal como se muestra en la primera imagen de los resultados. Mediante un bucle **`for`**, se aplicó la función **`hough_line()`** a pequeñas imágenes individuales correspondientes a cada triángulo. Estas imágenes fueron previamente obtenidas utilizando el etiquetado realizado en pasos anteriores. Posteriormente, se utilizó la función **`hough_line_peaks()`** para identificar los puntos más representativos de la transformación, es decir, aquellos que indican la presencia de líneas. Estos parámetros resultantes fueron utilizados para comprobar visualmente la precisión de las líneas detectadas, lo cual se muestra en las primeras imágenes de los resultados. También cabe remarcar, se transformaron los ángulos que teníamos

originalmente de radianes a grados. Note que tuvimos problemas con el primer triángulo, hasta la fecha desconocemos el por qué, pero lo arreglamos usando algunas técnicas para calcular los valores para esa línea en específico, si quiere saber más al respecto, los comentarios en el código son una buena fuente de explicación a fondo.

4. (2 puntos) Ahora elija del conjunto de segmentos de rectas aquellas que forman cada triángulo, haga una tabla donde haya 3 columnas, cada una con el número de segmento de recta (del inciso 3) que forman un triángulo, los renglones representan a cada triángulo. (ejemplo, supongamos que las aristas A1, A2 y A3 forman el triángulo T1 y las aristas A8, A17 y A18 forman al triángulo T6, entonces la tabla quedará de la siguiente manera:

Resultados:

Triangulo	Arista 1	Arista 2	Arista 3
T0	A1	A2	A3
T1	A4	A5	A6
T2	A7	A8	A9
T3	A10	A11	A12
T4	A13	A14	A15
T5	A16	A17	A18

Explicación:

En este apartado, parece ser que no hay que hacer mucha explicación, sencillamente fue agrupar cada Arista con tu triángulo con las funciones de pandas, específicamente con la función “**pivot_table**”. Note por favor, que en nuestro caso los triángulos y las aristas están en orden ascendente, por lo tanto, el triángulo 0 contiene las tres primeras aristas, el 1 contiene las siguientes tres, y así para los 6 triángulos.

5. (2 puntos) Encontrar las aristas que son paralelas entre triángulos diferentes. Reporte cuantos grupos de aristas paralelas encontró e indique cómo están conformados estos grupos.

Resultados:

Tabla de resultados, aquellas aristas en la misma fila son aproximadamente paralelas

Grado (aprox)	Arista 1	Arista 2	Arista 3
-20.0	A2	A6	
-30.0	A3	A9	A18
50.0	A5	A11	A14
-90.0	A7	A16	
30.0	A8	A17	

Explicación:

En este caso parece que solo hay dos líneas EXACTAMENTE paralelas, una del triángulo 2 y una del triángulo 5 con valor -90, sin embargo, parece ser que si nos ponemos menos exigentes y en lugar de requerir que los ángulos sean exactamente iguales, añadimos un pequeño márgen de error (debido a que los métodos que utilizamos tienen fallas o errores pequeños) podremos obtener mejores resultados. Se propuso un margen de error de 5 grados. **En este caso hubo 5 grupos de líneas paralelas conformadas por aquellas que se ven en la tabla de los resultados.**

PREGUNTAS

1. Conociendo la ecuación de las rectas y la longitud de los segmentos que forman cada arista de los triángulos, ¿se puede calcular el área del triángulo? De ser cierta la aseveración, ¿el área coincide con lo encontrado en el objetivo 1?

Sí, de hecho esto se hace con la fórmula de Herón, primero tenemos que calcular el semiperímetro, después de eso aplicar la siguiente fórmula:

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

Esto nos da como resultado lo siguiente:

Triangulo	Calculated_Area	Area	Diff_Area
T0	15442.0	15013	429.0
T1	11431.0	11491	60.0
T2	10995.0	10936	59.0
T3	4593.0	4664	71.0
T4	21068.0	20913	155.0
T5	11141.0	11042	99.0

La primera columna es el área que calculamos con la fórmula de Herón, la segunda la que se calculó con regionprops_table y la tercera es la variación, o diferencia que hay entre una y otra. Note que, como era de esperarse, el triángulo 0 es el que más problemas nos dio, por tanto, es el que tiene más diferencias en total. Todos los demás varían solamente en 60, 70, 100 y 150 píxeles, aunque parece que 100 ya es mucho, creo que el resultado es satisfactorio debido a las aproximaciones que hace la transformación de Hough. Creo que son buenos resultados, exceptuando por el 0, una lástima.

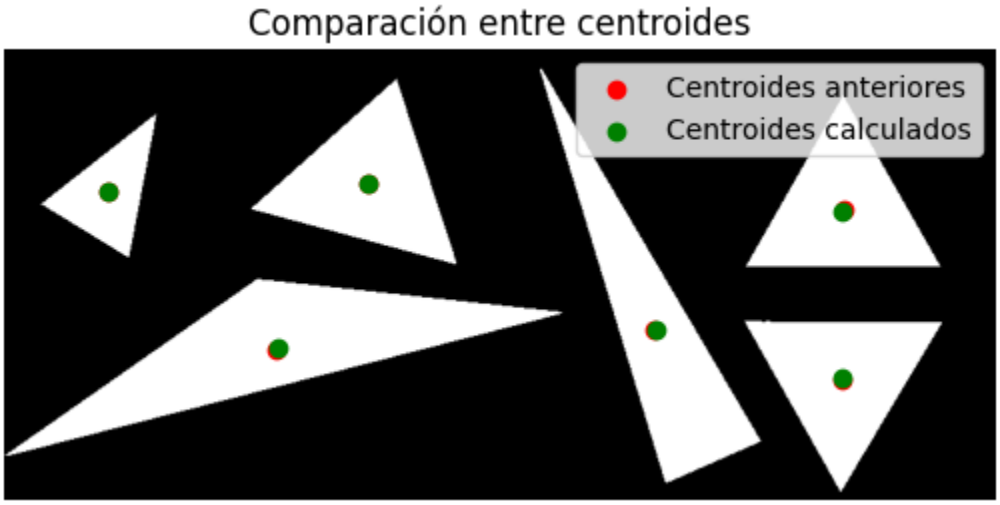
2. Conociendo la ecuación de las rectas y la longitud de los segmentos que forman cada arista de los triángulos, ¿se puede calcular el centroide del triángulo? De ser cierta la aseveración, ¿el centroide coincide con lo encontrado en el objetivo 1?

Sí, se pueden calcular a partir de los puntos medios de cada una de las rectas, en una parte previa del código calculamos las coordenadas de cada uno de los vértices para cada uno de los triángulos, eso significa que usando estos vértices, será más sencillo obtener los centroides:

Triangulo	Calculated_Centroid_Col	Centroid_Col	Col_Diff	Calculated_Centroid_Row	Centroid_Row	Row_Diff
T0	527.904695	527.575368	0.329327	225.850595	225.923600	0.073005
T1	294.882577	295.009747	0.127170	108.804466	108.808024	0.003558
T2	680.118617	680.500000	0.381383	130.000000	129.330468	0.669532
T3	84.634160	84.635506	0.001346	115.083445	114.913593	0.169852
T4	222.652833	219.636877	3.015956	241.740867	242.733515	0.992648
T5	680.155025	679.994113	0.160912	266.295797	266.330647	0.034850

La primera columna es el centroide respecto al eje x, o columnas, que calculamos, la segunda es la que habíamos calculado anteriormente, la tercera es la variación, o diferencia que hay entre una y otra, la cuarta, la quinta y la sexta son los mismos pero para las filas. NOTE que aquí sí es MUY similar, lo cual es una buena señal, incluso

podemos graficar la comparativa entre centroides calculados y centroides obtenidos anteriormente.



La diferencia tan notoria en las áreas puede darse a que ligeros cambios en inclinación de las líneas resultan en grandes cambios en el área, pero no grandes cambios en los centroides, por eso no existe un cambio significativo en estos.