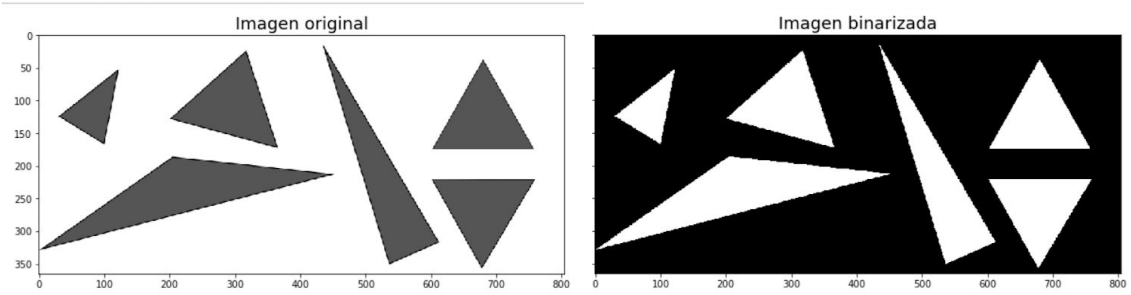
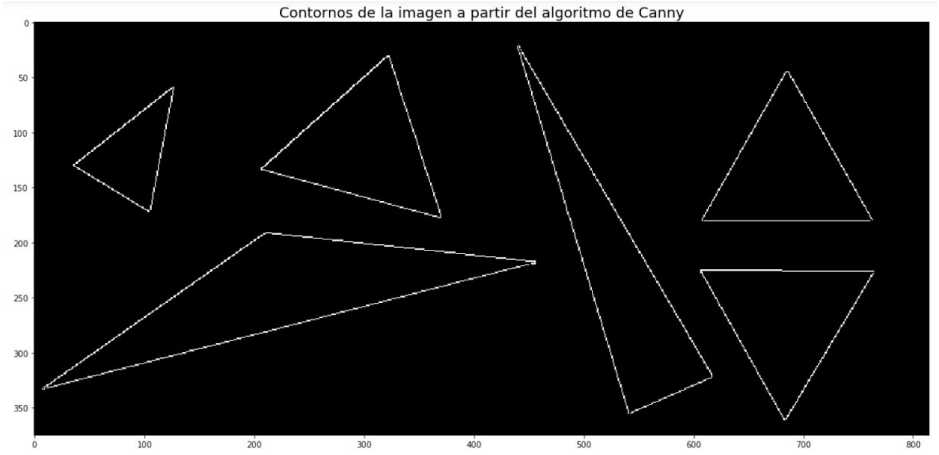
Reporte del Tema 2. Segmentar triángulos de una imagen

Resultados de la primera imagen.

Para la primera imagen, se eligió el umbral para la binarización a ojo, después de observar el histograma de intensidades. Así, todos los pixeles con intensidad menor a 170 se mandaron a blanco y el resto a negro. Se muestra el resultado:

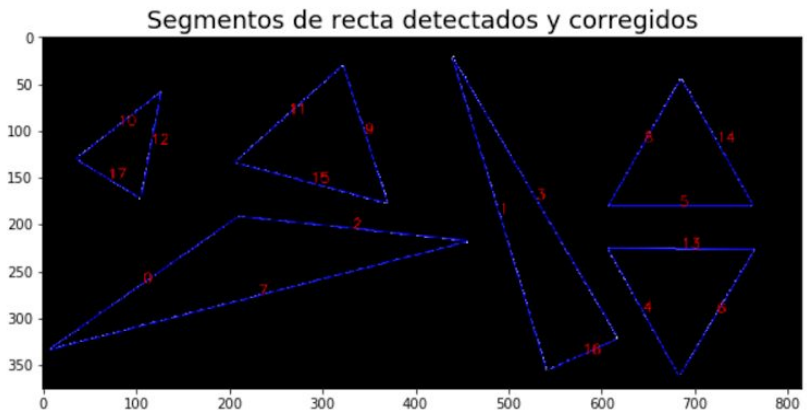


Para segmentar las aristas de los triángulos, se añadió un padding de 5 pixeles a cada lado de la imagen y al resultado se le aplicó el algoritmo de Canny con umbrales 50 y 200. La razón de estos valores es que son los que se incluyen en el código de ejemplo de la documentación de openCV y no se cambiaron porque entregaron buenos resultados.



La imagen anterior es la solución al punto 1.

Para obtener los modelos de los segmentos de recta que forman las aristas de los triángulos, se aplicó el método del espacio de Hough. Cada línea obtenida se dibujó sobre la imagen de las aristas junto con una etiqueta. Posteriormente, se eliminaron manualmente las rectas redundantes y se alargaron aquellas que lo necesitaban. El resultado fue el siguiente:



Debido a que la función de Python utilizada para obtener las rectas devuelve solo las coordenadas de los pixeles de sus extremos, los parámetros ρ y θ se calcularon con las fórmulas:

$$\theta = arctan(\frac{x_1 - x_2}{y_2 - y_1})$$

$$\rho = x_1 cos(\theta) + y_1 sen(\theta)$$

$$longitud = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

donde (x_1, y_1) y (x_2, y_2) son las coordenadas dadas por la función de Hough. A continuación, se muestran los resultados redondeados:

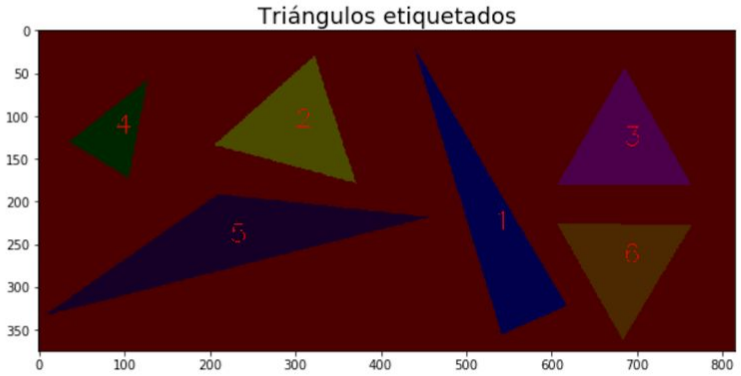
segmento	theta	rho	longitud
0	55	277.0	246
1	-17	415.0	348
2	-84	-168.0	246
3	-31	367.0	339
4	-29	420.0	156
5	-90	-180.0	153
6	31	773.0	154
7	76	325.0	462
8	30	616.0	155
9	-18	298.0	152
10	52	124.0	115
11	48	237.0	153
12	11	136.0	113
13	-90	-221.0	156
14	-30	572.0	155
15	-75	-77.0	169
16	66	545.0	81
17	-59	-94.0	78

La tabla anterior es el resultado del punto 2.

Para obtener las rectas paralelas, se utilizó un algoritmo encontrado en GeeksForGeeks para hallar pares iguales en un conjunto de datos. En este caso, para hallar pares de rectas con el mismo ángulo. Debido a la falta de tolerancia en el algoritmo, el único par de rectas paralelas encontrado fue el (5, 13), aunque probablemente la intención del dibujante también era hacer paralelos los pares (4, 14) y (6, 8). Lo anterior es el resultado del punto 3.

Para segmentar los triángulos, se utilizó la función *label* de scikit-image. Esta función devuelve un arreglo del tamaño de la imagen en donde los pixeles de cada

región disjunta tiene el mismo número, distinto del de las otras regiones, y los pixeles que no pertenecen a ninguna región son cero. La función se aplicó a la imagen binarizada y se dibujaron las etiquetas, como se muestra a continuación:



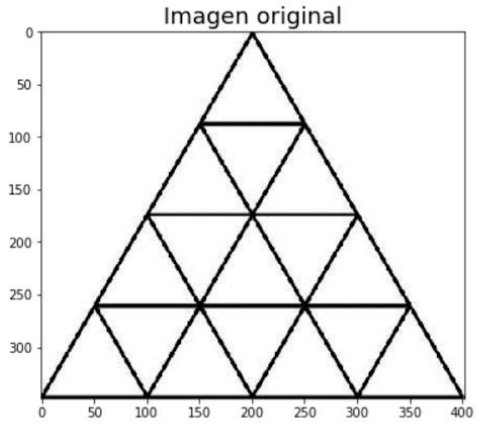
Para hallar las rectas pertenecientes a cada triángulo se verificó, para cada recta, si alguno de sus pixeles extremos quedaba dentro de alguna región etiquetada. Si no, se calculaban dos puntos, uno un poco a la derecha y otro un poco a la izquierda del punto medio de la recta, y se comprobaba si alguno de ellos pertenecía a alguna región y a cuál. Los resultados del punto 4 se resumen en la siguiente tabla:

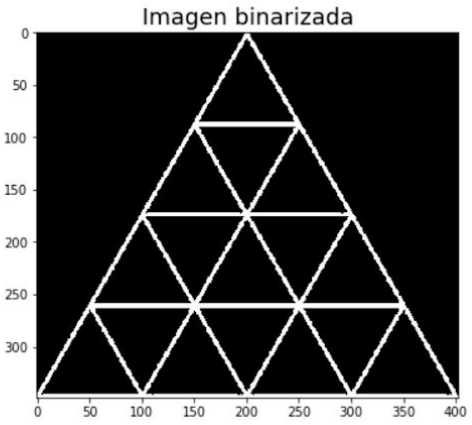
triángulo	recta1	recta2	recta3
1	1	3	16
2	9	11	15
3	5	8	14
4	10	12	17
5	0	2	7
6	4	6	13

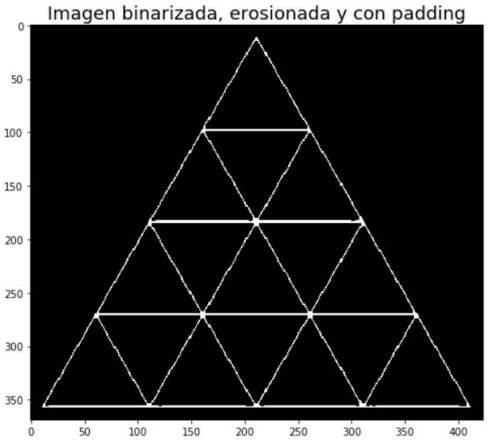
Los datos mostrados en la tabla son consistentes con los mostrados en las imágenes de las rectas y los triángulos etiquetados.

Resultados de la segunda imagen.

Para segmentar las aristas de los triángulos de la segunda imagen, se aplicó el mismo umbral que con la imagen 1 para la binarización, se añadió un padding de diez pixeles a cada lado de la imagen binarizada y se erosionó la imagen obtenida con un kernel 3x3 de puros 1’s. A continuación, se muestran los resultados:

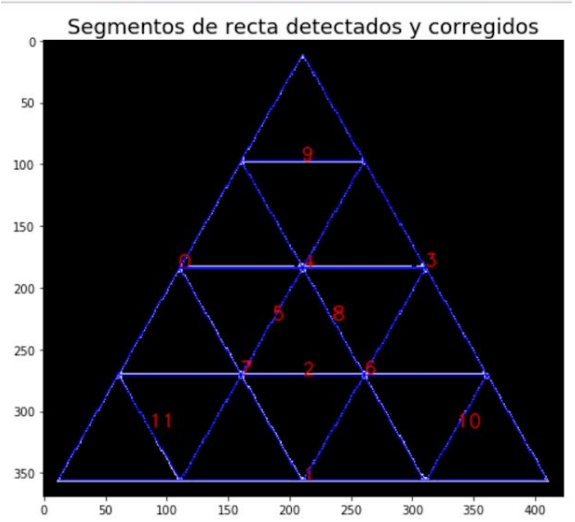






La imagen anterior es el resultado del punto 1.

Para obtener los modelos de rectas, se realizó el mismo procedimiento que con la primera imagen: aplicar el método de Hough, limpiar rectas redundantes a mano y calcular los parámetros a partir de las coordenadas de los extremos. Las siguientes imagen y tabla resumen los resultados del punto 2:



segmento	theta	rho	longitud
0	30	188.0	397
1	-90	-356.0	398
2	-90	-271.0	300
3	-30	177.0	397
4	-90	-184.0	200
5	30	275.0	298
6	30	361.0	196
7	-30	4.0	196
8	-30	91.0	297
9	-90	-97.0	97
10	30	448.0	96
11	-31	-88.0	97

Para obtener las rectas paralelas también se utilizó el mismo procedimiento que con la primera imagen, obteniéndose lo siguiente:

```
Hay 3 conjuntos de líneas paralelas.
Las líneas [0, 5, 6, 10] son paralelas entre sí.
Las líneas [1, 2, 4, 9] son paralelas entre sí.
Las líneas [3, 7, 8] son paralelas entre sí.
```

Solo se presentó un falso negativo: la recta 11 también es paralela a las rectas 3, 7 y 8. En la tabla anterior, se muestra que la diferencia es de tan solo un grado. El ángulo para la recta 11 se modificó a mano porque era necesario para resolver el último punto.

Resolver el punto 3 fue más complicado para esta imagen que para la anterior. Primero, se guardaron todas las combinaciones posibles de tres de las rectas detectadas en la imagen. Se obtuvieron 220. Segundo, se eliminaron todas las combinaciones que incluyeran pares de líneas paralelas. Quedaron cerca de 65. Finalmente, se encontraron los puntos de cruce entre las rectas de las combinaciones restantes. Cada conjunto de tres rectas se tomaba como válido para formar un triángulo si la distancia entre sus cruces era mayor a 90 (ya que los segmentos de recta más pequeños detectados en la imagen miden aproximadamente 96 pixeles) y si los cruces no quedaban fuera del contorno formado por las rectas 0, 1 y 3. Para cada triángulo válido se dibujó un polígono con los vértices encontrados y se guardó una imagen con este polígono superpuesto en gris. A continuación, se muestran los resultados junto con una imagen muestra:

```
triángulo       rectas
0         [0 1 3]
1         [0 1 7]
2         [0 1 8]
3         [ 0  1 11]
4         [0 2 3]
5         [0 2 7]
6         [0 2 8]
7         [0 3 4]
8         [0 3 9]
9         [0 4 8]
10        [1 3 5]
11        [1 3 6]
12        [ 1  3 10]
13        [1 5 7]
14        [1 5 8]
15        [1 6 8]
16        [2 3 5]
17        [2 3 6]
18        [2 5 8]
19        [ 2  5 11]
20        [2 6 7]
21        [ 2  8 10]
22        [3 4 5]
23        [4 5 7]
24        [4 6 7]
25        [4 6 8]
26        [5 8 9]
En total hay 27 triángulos en la imagen
```