Proyecto - Cálculo de diámetro arterial

Ramon Orlando Ruiz Olais

26/02/2024

1 Introducción

Determinar el diámetro arterial a través de métodos computacionales requiere realizar múltiples pasos de procesamiento de imágenes para obtener un resultado preciso. También se requieren varias fotografías que nos permitan utilizar la mayor parte de ellas como un conjunto de entrenamiento, para determinar los parámetros más efectivos en la mejora de las imágenes.

Las fotografías deben ser convertidas a escala de grises, y deben ser convertidas a negativos, esto para aplicar un método de mejora de imágenes, en el presente trabajo se utilizó TopHat White para esto, que consiste en restar a la imagen en negativo su apertura (la apertura consiste en erosionar y después dilatar la imagen de entrada).

Una vez se hayan mejorado las imágenes se segmentan para que queden en blanco y negro únicamente, de este modo buscamos distinguir el objeto de interés del fondo (la arteria). Se realiza un calculo de la mayor componente conexa en la imagen para eliminar el ruido de fondo y calculamos los pixeles que pasan por en medio de la arteria (a esto le llamamos esqueleto). Finalmente podemos utilizar el esqueleto y la imagen umbralizada para determinar cuál es el ancho arterial en distancia de pixeles, calculando cuál es la distancia del esqueleto a el fondo negro de la imagen.

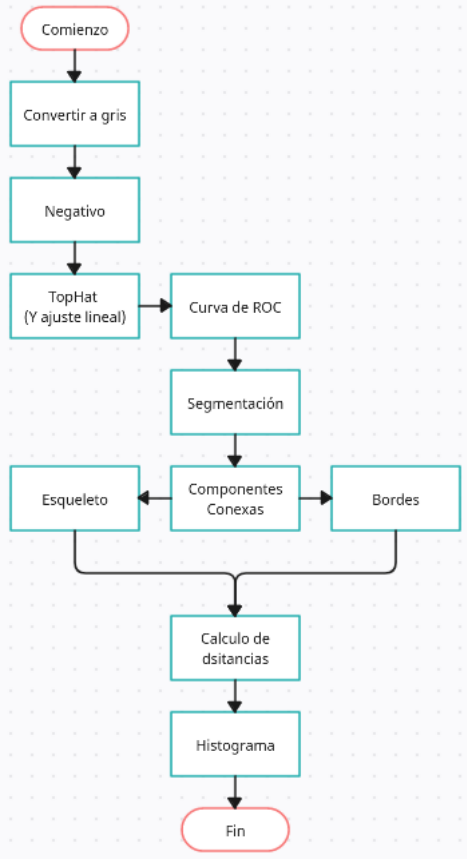

Figure 1: Diagrama de flujo general del proyecto.

2 Curva de ROC para escoger los mejores parámetros.

Para mejorar las imágenes con tophat antes de segmentarlas, podemos escoger distintos elementos estructurantes de distintos tamaños, entonces, para comenzar, es necesario determinar cuáles serán los mejores parámetros para mejorar las imágenes y así obtener mejores resultados en nuestra segmentación.

Para esto realicé un algoritmo iterativo que aplica Tophat a 100 imágenes de prueba con un elemento estructurante y un tamaño dado, luego las concatena y calcula los valores de sensibilidad y especificidad y repite para cada distinto elemento estructurante y tamaño, al final toma los valores guardados en archivos de texto e integra con método del trapecio, al final el algoritmo devuelve el elemento estructurante y tamaño correspondientes a la integral más grande.

```
El valor mayor de Az es: 0.918374
Parámetros: Disco | Tamaño 9 (19x19)
Tiempo transcurrido: 4627.67 segundos.
```

Figure 2: Resultados del algoritmo que calcula las curvas de ROC.

El algoritmo tarda un aproximado de 1 hora con 28 minutos en terminar de ejecutarse, y determinó que el mejor elemento estructurante para este conjunto de imágenes es el disco con un tamaño de 19x19.

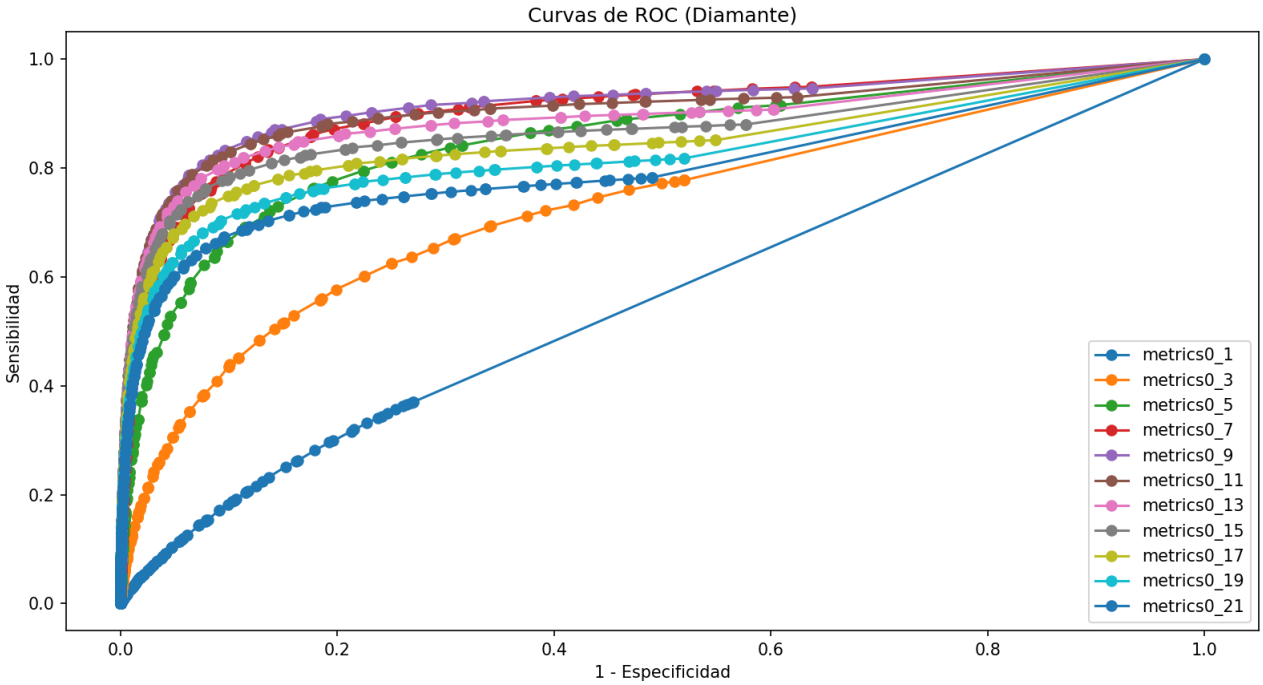

Figure 3: Curvas de ROC para los elementos con estructura de diamante.

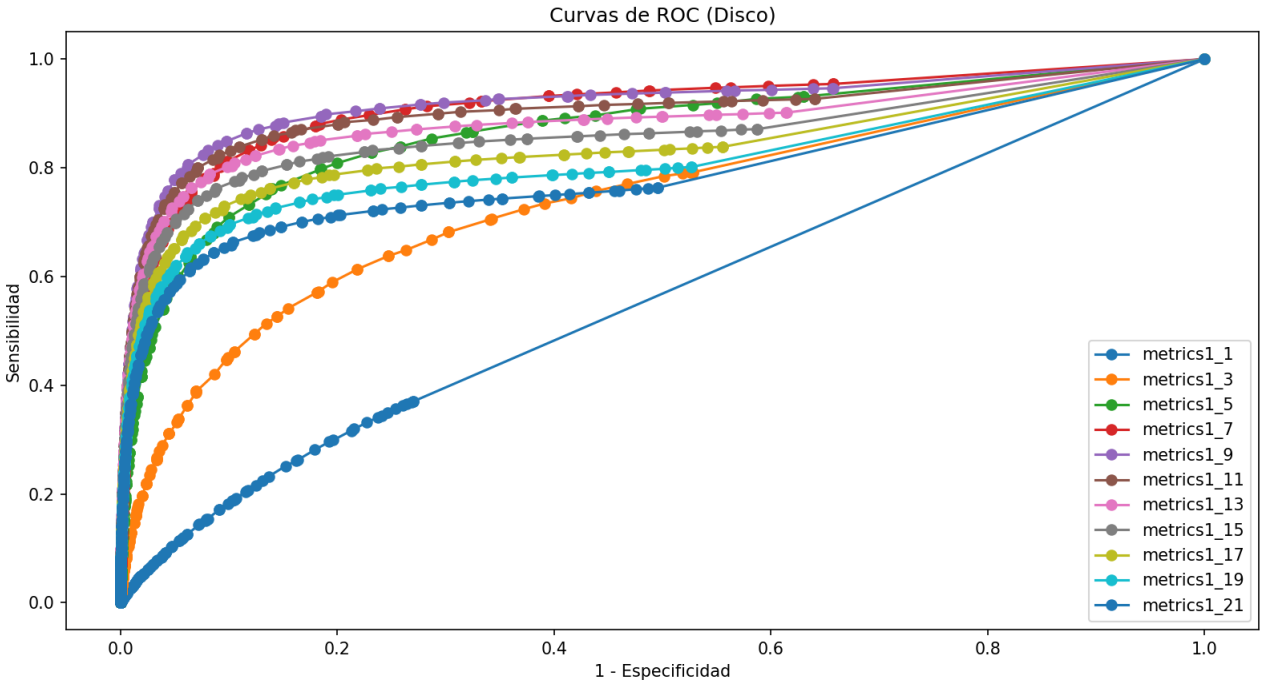

Figure 4: Curvas de ROC para los elementos con estructura de disco.

Algorithm 1 Curvas de ROC

for $interruptor = 0$ **to** 1 **do**
  for $j = 1$ **to** 21 **step** 2 **do**
    $n = 2 \times j + 1$
    Creamos los vectores para concatenar las imágenes
    for $i = 1$ **to** $numImages$ **do**
      Se declaran los nombres de las imágenes a cargar
      Se cargan las imágenes
      Sacamos negativo de las imágenes
      Aplicamos TopHat
      Calculamos los valores máximos y mínimos de intensidad
      Realizamos un ajuste lineal
      Se agregan las imágenes a un vector
    end for
    Se concatenan las imágenes de los vectores
    Se calculan los valores de sensibilidad y especificidad para cada elemento.
    Se guardan los resultados.
  end for
end for

3 Resultados de TopHat con los mejores parámetros.

Una vez obtenidos los mejores parámetros tomamos alguna de las imágenes restantes de nuestro conjunto de datos. Tomando por ejemplo la número 102 y aplicando Top Hat con elemento estructurante con forma de disco de tamaño 19x19 obtenemos lo siguiente:

Figure 5: Imagen original

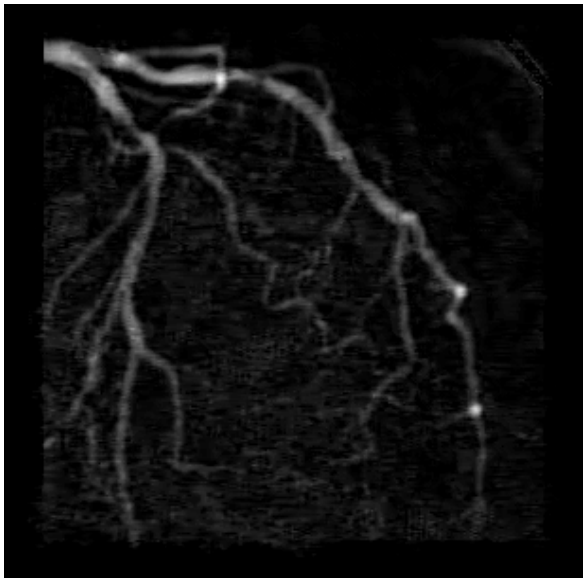

Figure 6: Imagen con Top Hat

4 Segmentación con Ridler & Calvard

El objetivo de la segmentación es convertir nuestra imagen en escala de grises en una imagen binaria (blanco y negro). Para esto existen diversos método y en el presente trabajo se empleó el método de Ridler&Calvard.

Algorithm 2 RidlerAndCalvard

1: **function** RIDLERANDCALVARD(input)
2: $binario =$ input
3: $sum = 0$
4: $tol = 0.0005$
5: **for** $y = 0$ **to** input.height **do**
6: **for** $x = 0$ **to** input.width **do**
7: $sum = sum +$ input.pixels$[y][x]$
8: **end for**
9: **end for**
10: mu $= sum/($input.height $\times$ input.width$)$
11: $new\_thresh =$ mu
12: $prev\_thresh = 0$
13: $threshold = 0$
14: $numN1 = 0$, $denN1 = 0$, $numB1 = 0$, $denB1 = 0$, $mN = 0$, $mB = 0$, $k = 0$
15: **while** true **do**
16: $numN1 = 0$, $denN1 = 0$, $numB1 = 0$, $denB1 = 0$, $mN = 0$, $mB = 0$, $k = 0$
17: $prev\_thresh = new\_thresh$
18: $threshold = prev\_thresh$
19: **for** $y = 0$ **to** input.height **do**
20: **for** $x = 0$ **to** input.width **do**
21: **if** input.pixels$[y][x] \leq$ threshold **then**
22: $numN1 = numN1 +$ input.pixels$[y][x]$
23: $denN1 = denN1 + 1$
24: **else**
25: $numB1 = numB1 +$ input.pixels$[y][x]$
26: $denB1 = denB1 + 1$
27: **end if**
28: **end for**
29: **end for**
30:

```
function RidlerAndCalvard2(input)
    mN = numN1/denN1
    mB = numB1/denB1
    new_thresh = (mB + mN)/(2)
    if |new_thresh − prev_thresh| ≤ tol then
        break
    end if
    print "threshold =" threshold

    for y = 0 to binario.height do
        for x = 0 to binario.width do
            if binario.pixels[y][x] ≤ threshold then
                binario.pixels[y][x] = 0
            else
                binario.pixels[y][x] = 255
            end if
        end for
    end for
    return binario
end function=0
```

Figure 7: Imagen binarizada con Ridler & Calvard

Los resultados del método de segmentación son algo indeseables pues deja huecos en múltiples zonas de la arteria e incluso elimina secciones casi completas.

5 Componentes conexas

Para eliminar el ruido de la imagen recién segmentada (el cuál se puede manifestar como puntos blancos inconexos en múltiples zonas de la imagen) se utiliza un algoritmo que determina cuál es el conjunto de pixeles blancos conexos más grandes y los re-dibuja sobre un fondo negro, así capturamos el cuerpo de interés y eliminamos el ruido.

Algorithm 3 MayorComponenteConexa

```
1: function MayorComponenteConexa(input)
2:     test = input
3:     output = input
4:     Crear una cola cola
5:     Crear un conjunto conjuntoPosicion
6:     Crear una variable frente
7:     Crear un vector totalConjuntos
8:     for y = 1 to test.height − 1 do
9:         for x = 1 to test.width − 1 do
10:            if test.pixels[y][x] = 255 then
11:                conjuntoPosicion es vacío
12:                cola.push((y, x))
13:                while not cola.empty() do
14:                    frente = cola.front()
15:                    if frente.first > 1 and frente.first < test.height − 1 and frente.second > 1 and
    frente.second < test.width − 1 then
16:                        if test.pixels[frente.first][frente.second + 1] then
17:                            cola.push((frente.first, frente.second + 1))
18:                            test.pixels[frente.first][frente.second + 1] = 0
19:                        end if
20:                        if test.pixels[frente.first][frente.second − 1] then
21:                            cola.push((frente.first, frente.second − 1))
22:                            test.pixels[frente.first][frente.second − 1] = 0
23:                        end if
24:                        if test.pixels[frente.first + 1][frente.second] then
25:                            cola.push((frente.first + 1, frente.second))
26:                            test.pixels[frente.first + 1][frente.second] = 0
27:                        end if
28:                        if test.pixels[frente.first − 1][frente.second] then
29:                            cola.push((frente.first − 1, frente.second))
30:                            test.pixels[frente.first − 1][frente.second] = 0
31:                        end if
32:                    end if
33:                    test.pixels[frente.first][frente.second] = 0
34:                    conjuntoPosicion.insert(frente)
35:                    cola.pop()
36:                end while
37:                totalConjuntos.push_back(conjuntoPosicion)
38:            end if
39:        end for
40:    end for
```

```
function MayorComponenteConexa2
    longitudMaxima = 0
    Crear un conjunto conjuntoMaximo
    for i = 0 to totalConjuntos.size() − 1 do
        if totalConjuntos[i].size() > longitudMaxima then
            conjuntoMaximo = totalConjuntos[i]
            longitudMaxima = totalConjuntos[i].size()
        end if
    end for
    for y = 0 to output.height − 1 do
        for x = 0 to output.width − 1 do
            output.pixels[y][x] = 0          ▷ Establece el valor del píxel en negro en la imagen de salida
        end for
    end for
    for cada pixel pix en conjuntoMaximo do
        y = pix.first
        x = pix.second
        output.pixels[y][x] = 255            ▷ Establece el valor del píxel en blanco en la imagen de salida
    end for
    return output
end function=0
```

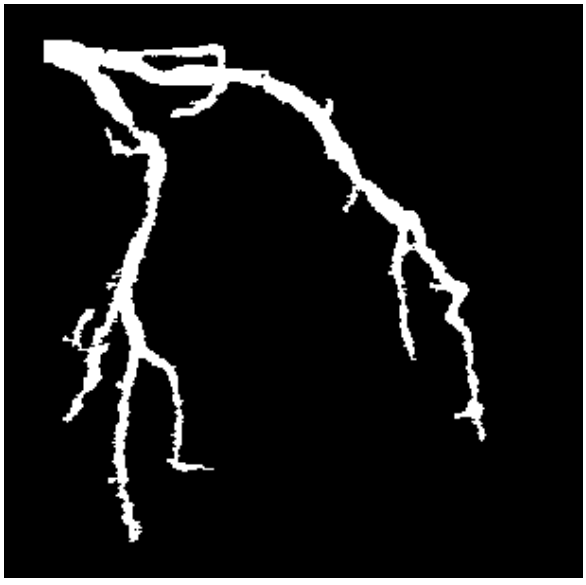

Figure 8: Imagen conexa

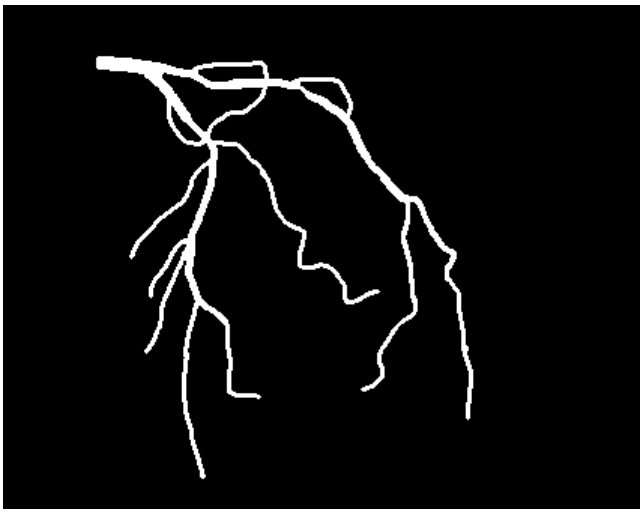

Figure 9: Imagen ground truth

Dado que la segmentación no fue perfecta es inevitable que existan huecos que prácticamente cortan el cuerpo de interés transversalmente, esto provoca que la componente conexa más grande no sea el cuerpo completo (aún si el corte es

de muy poco grosor), por esto obtuvimos el resultado de la figura 7.

6 Cálculo del esqueleto

Para determinar el grosor de la arteria es necesario determinar cuáles son los pixeles que pasan por el centro de la misma, a esto le llamamos el "esqueleto" y podemos obtenerlo utilizando el algoritmo de Zhang-Suen.

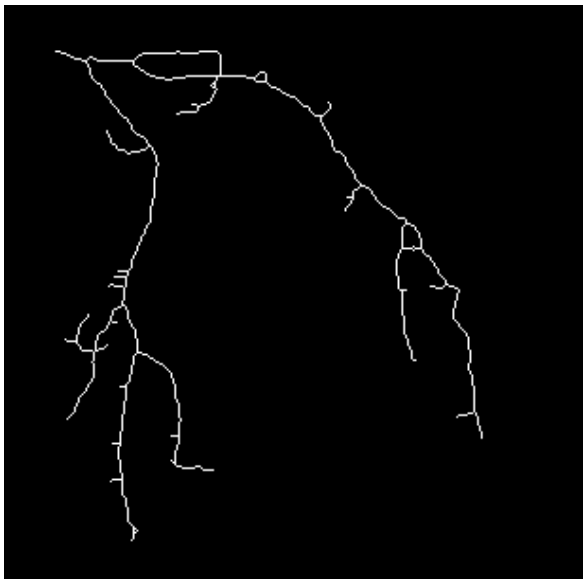

Figure 10: Esqueleto de la arteria

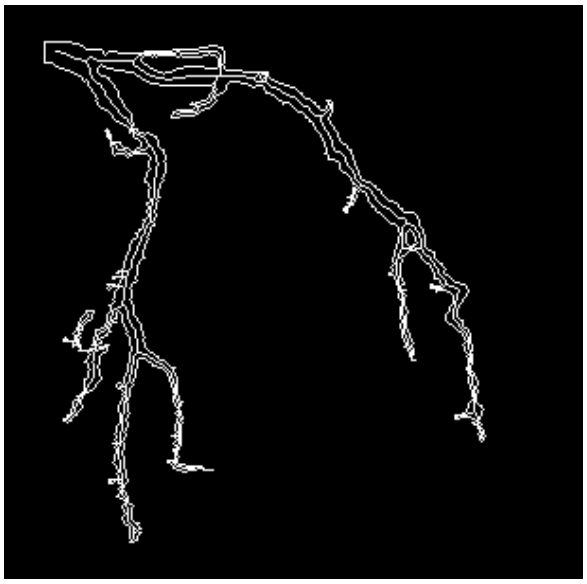

Figure 11: Suma de la imagen original

En general, se consiguió obtener un objeto de grosor de un píxel que pasa aproximadamente por el centro de la arteria original.

7 Cálculo del diámetro arterial

Finalmente, haciendo uso del esqueleto obtenido y la imagen umbralizada podemos calcular el grosor de la misma buscando la distancia de los pixeles que pasan por el centro hasta la zona donde los pixeles se vuelven de color negro (fuera de la arteria en la imagen segmentada).

Algorithm 4 Diameter

```
function DIAMETER(inputOG, inputSke)
    D = 0
    interruptor = 0
    Crear un vector distancias
    for y = 0 to inputSke.height − 1 do
        for x = 0 to inputSke.width − 1 do
            if inputSke.pixels[y][x] = 255 then
                D = 1
                interruptor = 0
                while interruptor = 0 do
                    for k = 0 to D − 1 do
                        if inputOG.pixels[y + k][x − (D − k)] = 0 then
                            interruptor = 1
                            break
                        end if
                        if inputOG.pixels[y − k][x − (D − k)] = 0 then
                            interruptor = 1
                            break
                        end if
                        if inputOG.pixels[y + k][x + (D − k)] = 0 then
                            interruptor = 1
                            break
                        end if
                        if inputOG.pixels[y − k][x + (D − k)] = 0 then
                            interruptor = 1
                        end if
                    end for
                    D = D + 1
                end while
                distancias.push_back(D − 1)
            end if
        end for
    end for
    return distancias
end function
```

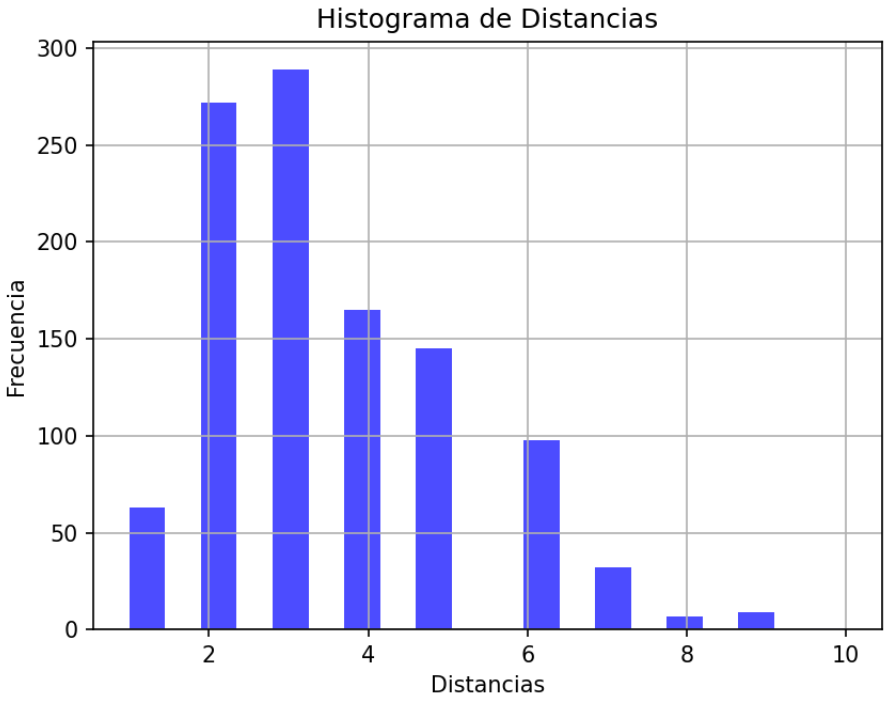

Figure 12: Histograma de distancias.

8 Conclusión

Se puede decir que se logró el objetivo general del proyecto, el cual fue determinar los mejores parámetros para mejorar las imágenes, posteriomente se mejoraron y segmentaron, luego se calculó la componente conexa más grande y el esqueleto de la arteria, para finalmente lograr determinar el diámetro de la misma, algunos de los algoritmos presentan resultados que pueden ser mejorados considerablemente, en particular, el algoritmo de mejora o el algoritmo de segmentación son los que presentan algunos problemas al utilizarlos fuera de las imágenes de prueba. Realicé un cálculo del valor Az para mis 34 imágenes finales concatenadas comparados con sus respectivos ground truth, y se obtuvo un valor de Az = 0.81 aproximadamente.