122-41

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Ciudad de México

División de Ingeniería y Arquitectura

Ingeniería en Electrónica y Comunicaciones
Departamento de Ingeniería Eléctrica y Electrónica

Segmentación y Caracterización en Angiografía Cardiaca: comparación de tres algoritmos

Autores: Miguel Angel Gálvez
Salvador Terán Fraga
Israel Muñoz González

Asesor: Dr. Jorge Eduardo Brieva

ITESM
CAMPUS CIUDAD DE MEXICO
BIBLIOTECA

México, D. F. Mayo 2004

~~Tesis~~ EGIA_ PROYE
RC691.6.A53
G35
2004

ÍNDICE GENERAL

TEMA **PÁGINA**

1 JUSTIFICACIÓN Y OBJETIVOS 2

1.1 INTRODUCCIÓN 2
1.2 JUSTIFICACIÓN 3
1.3 OBJETIVOS 3

2 LA ANGIOGRAFÍA CARDIACA 5

2.1 TÉCNICA 5
2.2 CARACTERÍSTICAS 6
2.3 APLICACIONES DE LA ANGIOGRAFÍA: SISTEMA DE VISIÓN PARA EL DIAGNÓSTICO Y LA CIRUGÍA ASISTIDO POR COMPUTADORA 6

3 DESCRIPCIÓN DEL SISTEMA PROPUESTO 8

3.1 LA SEGMENTACIÓN 9
3.1.1 Wavelets 9
3.1.2 Contornos Deformables 14
3.1.3 Momentos Geométricos 21
3.2 VALIDACIÓN DE MÉTODOS 27
3.3 HERRAMIENTAS COMPUTACIONALES DE AYUDA AL DIAGNÓSTICO 31
3.3.1 Detección de Bordes 31
3.3.2 Estimación de Diámetros aparentes 31
3.3.3 Cuantificación de severidad de la estenosis 32

4 RESULTADOS 33

4.1 SEGMENTACIÓN UTILIZANDO EL ALGORITMO DE WAVELETS 33
4.2 SEGMENTACIÓN UTILIZANDO EL ALGORITMO DE MOMENTOS GEOMÉTRICOS DE UNA IMAGEN 35
4.3 SEGMENTACIÓN UTILIZANDO ALGORITMO DE DEFORMACIÓN MEDIANTE B-SPLINES 38
4.4 ESTIMACIÓN DE ESTENOSIS APARENTE 39

5 CONCLUSIONES Y PERSPECTIVAS 42

6 FUENTES 44

7 ANEXOS 46

ANEXO A 46
ANEXO B 48
ANEXO C 54
ANEXO D 61
ANEXO E 65
ANEXO F 69

1 Justificación y Objetivos

1.1 Introducción

Las imágenes por rayos X fueron el primer medio de exploración de las estructuras internas del cuerpo y, durante mucho tiempo, fueron la única fuente de exploración de las estructuras internas del cuerpo humano. Ocupan un lugar muy importante en el diagnóstico de lesiones en los huesos y de tumores, así como, en un contexto de monitoreo poblacional, en el diagnóstico de enfermedades pulmonares. Actualmente son comúnmente utilizadas en el diagnóstico de cáncer de seno (mamografía) y de enfermedades vasculares (angiografía) en los diferentes órganos (cerebro, corazón) tanto a nivel arterial como venoso.

La angiografía digital, además de su utilidad en el diagnóstico de enfermedades vasculares, es una herramienta indispensable en la cirugía cerebral y cardiaca. Es el caso en la cirugía cerebral donde para una simple biopsia hay que evitar, hasta donde sea posible, tocar las principales arterias y venas. Igualmente es el caso en intervenciones cardiacas para la reducción y eliminación de estenosis.

En el contexto de diagnóstico y cirugía, la creación de herramientas computacionales que ayuden al cirujano especializado a mejorar su diagnóstico y su desempeño durante la operación es de gran importancia. La implementación de herramientas computacionales tiene como objetivo, de largo plazo, una automatización de las cirugías vasculares.

Un proceso de automatización cualquiera cuenta con diferentes módulos especializados. En este trabajo, nos interesaremos en dos de las componentes de cualquier sistema de visión: la segmentación y la descripción de objetos.

Este documento se organiza en cinco capítulos, el primero es introductorio y presenta la justificación y los objetivos. El segundo presenta la técnica de angiografía y las características de las imágenes angiográficas que serán el objeto de procesamiento en este trabajo. El tercero incluye una descripción del sistema propuesto de implementación, así como la explicación de los algoritmos implementados y la validación de los mismos. El cuarto muestra los resultados obtenidos en la realización de este proyecto. El quinto capítulo presenta las conclusiones a las que hemos llegado con la realización del

presente trabajo. Finalmente, se incluyen las referencias que hemos utilizado en la elaboración de nuestro proyecto y los anexos del mismo.

1.2 Justificación

En México, la cardiopatía coronaria es la patología cardiovascular dominante, un problema de prevención y control. Las estadísticas generales informan que las enfermedades del corazón figuran en primer lugar como causa de muerte desde hace más de 20 años. El problema, ahora se agrava gradualmente, al ritmo de entonces y sin indicio de control.

La arteriosclerosis en todas sus formas es responsable de por lo menos la cuarta parte de todos los fallecimientos del país. Los datos combinados de mortalidad y morbilidad, general y hospitalaria, indican que la letalidad hospitalaria llega a ser de 25% por infarto agudo de miocardio (IAM). La atención, antes de morir, la reciben sólo 8 de cada 100, testimonio del certificado de defunción. El restante 92% no acude a ningún hospital, por algún motivo. La incidencia anual de IAM se estima en 140,000 casos; considerando que por cada fallecido (35,000) sobreviven 3.

Las tasas de mortalidad por IAM, estandarizadas por edad, son más altas que EEUU y Canadá, lo cual crea una expectativa de mayor preocupación. La explicación más plausible sobre la situación es la deficiencia de la atención preventiva, misma que se debe incluir en las situaciones de urgencia y antes de llegar al hospital. Las formas agudizadas de hipertensión arterial, accidentes cerebrovasculares y diabetes mellitus comparten la misma suerte.

La atención temprana y adecuada para estas condiciones críticas, puede contribuir importantemente a disminuir la mortalidad cardiovascular del país, a corto y mediano plazo. El enfoque preventivo debe considerarse tanto en lo crónico como en lo urgente y lo crítico. Conviene incluir en el fomento a la salud la cultura por un corazón saludable y paralelamente promover una conciencia colectiva acerca de los problemas sobreagudos. La prestación de cuidados oportunos salva vidas y miocardios. El refuerzo a la atención pre-hospitalaria reducirá el tiempo de retraso en la llegada de casos con IAM. Llevar a cabo estas actividades es compatible con los programas existentes de reanimación cardiopulmonar, cadena de supervivencia y capacitación a técnicos en urgencias médicas.

1.3 Objetivos

Se busca realizar un estudio comparativo de algunas técnicas de segmentación aplicadas a las imágenes angiográficas coronarias. Se desea seleccionar e implementar métodos con las siguientes características:

- Robusto

- Con pocos parámetros para controlar
- Confiable
- Que sea una base para ayudar en el diagnóstico pre-operatorio, en la cirugía asistida por computadora y en el control post-operatorio.

2 La Angiografía Cardiaca

2.1 Técnica

La angiografía coronaria es un examen de los vasos sanguíneos o las cámaras del corazón. Un tubo muy delgado, llamado catéter, se inserta por un vaso sanguíneo en el brazo o la ingle (Figura 1). La punta del catéter se posiciona ya sea en el corazón o al inicio de las arterias que llegan al corazón, y un líquido especial, llamado medio de contraste, se inyecta. Este fluido es visible en las placas de rayos X y las imágenes que se obtienen se llaman angiogramas.

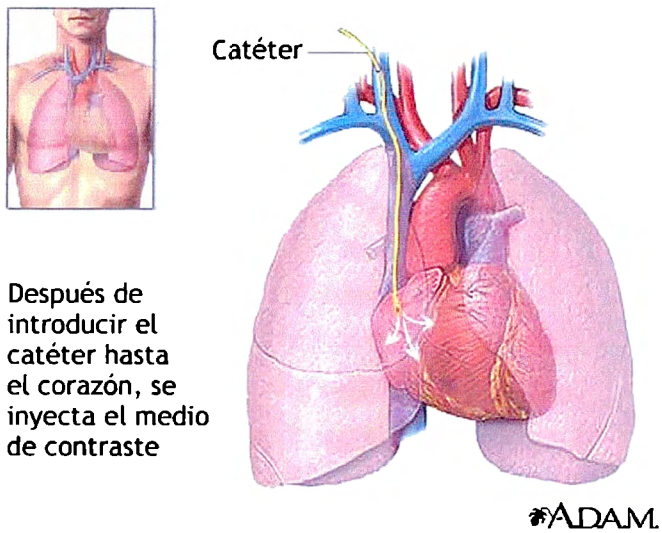


Figura 1. Cateterización cardiaca [17]

2.2 Características

- Resolución espacio-temporal: determina el nivel de detalle de los objetos y depende de la resolución de los detectores igualmente que del medio de contraste que va a permitir de diferenciar las estructuras de interés. La resolución de la angiografía digital es superior a otras técnicas de imágenes como el scanner o la resonancia magnética. La resolución temporal (decenas de imágenes por segundo) da una mejor información que otras técnicas.

- Las imágenes son de proyección es decir que dan una información en dos dimensiones solamente. Los ángulos de observación son escogidos de manera general según los dos puntos de vista clásicos: oblicuo anterior izquierdo y oblicuo anterior derecho.

- Es una técnica invasiva tanto desde el punto de vista de radiación como por medio del líquido de contraste. El medio de contraste está directamente relacionado con la calidad de la imagen y la coordinación entre su inyección y la adquisición de la imagen definirá la calidad de una angiografía.

- El costo de esta técnica es muy bajo comparado con otras técnicas de diagnóstico por imágenes.

- Es una herramienta confiable para el diagnóstico de estenosis y el grado de bloqueo de los vasos.

2.3 Aplicaciones de la Angiografía: Sistema de Visión para el Diagnóstico y la Cirugía Asistido por Computadora

La simulación y la asistencia en la dirección de movimientos del cirujano en intervenciones quirúrgicas a partir de imágenes obtenidas en la etapa de diagnóstico forman parte de las áreas de investigación de actualidad en biomédica. Un sistema de visión confiable es necesario para, por un lado, caracterizar el diagnóstico y, por otro lado, realizar la correspondencia de datos entre las imágenes de diagnóstico y las imágenes durante la operación. Esta correspondencia entre imágenes tiene el fin de guiar al médico en la intervención. Un sistema de visión está compuesto por los siguientes módulos:

- Formación de la imagen

- Detección-segmentación

- Formación de objetos relevantes

- Caracterización

- Visualización
- Interpretación de datos

En este proyecto nos localizaremos en la etapa de segmentación. Los métodos utilizados deberán basarse en el conocimiento de las características de las estructuras que queremos detectar. Aquí nombramos algunas de esas características:

- Se trata de estructuras en forma de cintas, equivalentes a proyecciones de objetos reales de forma cilíndrica.
- Las estructuras tienen bifurcaciones así como posibles cruces entre ellas.
- El reflujo del medio de contraste origina que, en algunas ocasiones, no existan uniformidades entre algunas áreas en la distribución del mismo y, por consiguiente, se detectan diferentes niveles de gris en las estructuras.
- Discontinuidades producidas por estenosis muy cerradas.
- Objetos con diferentes diámetros.
- Objetos intrínsecos como suturas realizadas anteriormente y extrínsecos como el catéter.

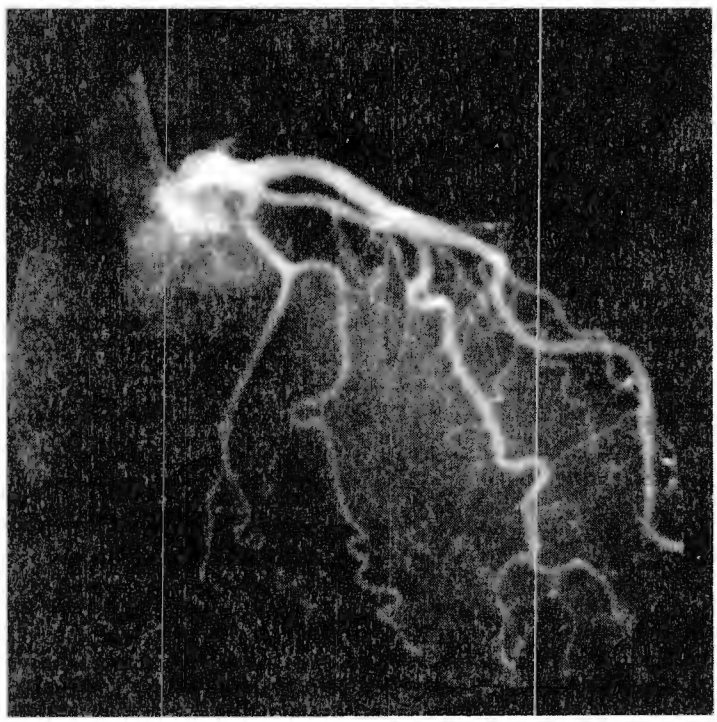


Figura 2. Imagen angiográfica

3 Descripción del Sistema Propuesto

El sistema propuesto pretende, por un lado, definir un protocolo de validación y comparación de métodos de segmentación y, por otro lado, desarrollar una plataforma de visualización y de implementación de herramientas de ayuda al diagnóstico. Los módulos de nuestro sistema son los siguientes:

- Segmentación de imágenes
- Un módulo de comparación y validación cualitativa de los algoritmos implementados mediante imágenes simuladas
- Un módulo de caracterización de patologías (estenosis), partiendo de una buena segmentación de las imágenes angiográficas

El sistema que proponemos es el que se muestra en la siguiente figura:

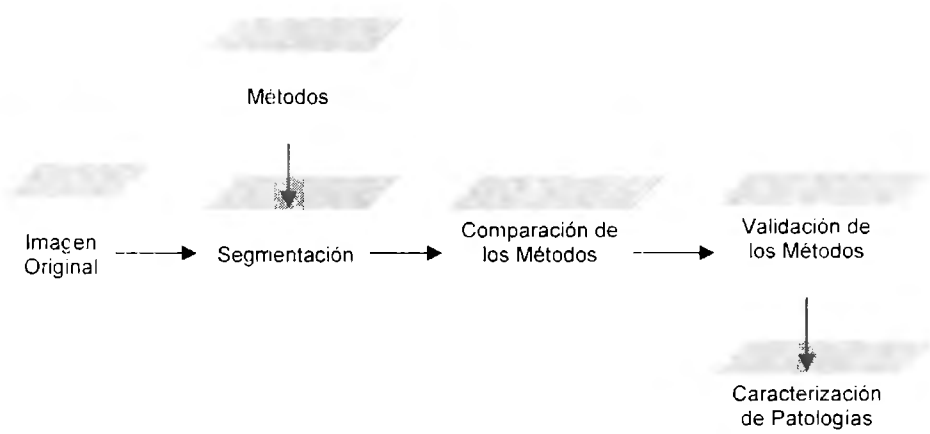


Figura 3. Sistema propuesto

3.1 La Segmentación

Se investigaron diferentes métodos de segmentación:

- Filtros FIR (wavelets, Gabor)
- Contornos deformables (snakes, splines)
- Operadores de detección (Sobel, momentos geométricos)

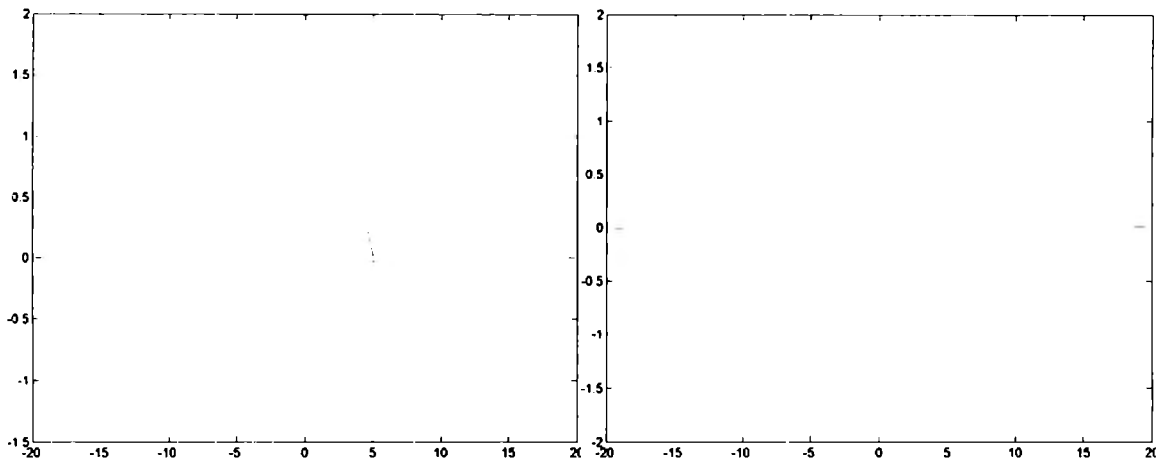
Se implementó en matlab cada uno de los algoritmos investigados y se realizó la segmentación de imágenes angiográficas por medio de los diferentes algoritmos.

3.1.1 Wavelets

Se asume que los vasos tienen un diámetro máximo, el cual será detectado por medio de esta descomposición. El análisis se lleva a cabo mediante una convolución con un filtro espacial unidimensional que tiene coeficientes definidos utilizando la siguiente función compleja [3]:

$$\Phi_{Ks}(x)=(1+\cos(2\pi F_0 x)).e^{-2i\pi kxF0x} \qquad (3.1)$$

si $|x| < 1 / 2F_0$ y $k_x \neq -1, 0, 1$ y k_x es entero



(a) (b)

Figura 4. Parte real (a) y parte imaginaria (b) de la función wavelet utilizada

Se muestrea (y se utiliza un ventaneo) de tal manera que, si 2N + 1 es el número total de coeficientes, N = INT ($1/2F_0$) (INT: parte entera) en donde N es el orden del filtro de wavelet.

Los parámetros del filtro son F_0 y k_x. Así, la función Φ_{Ks} define un filtro paso-banda con frecuencia central kF_0 y con una anchura proporcional a F_0. El valor de k_x, el cual establece la selectividad del filtro, se estableció como 2. La elección de F_0 está en función

del diámetro de los vasos (entre 5 y 30 pixeles). Para poder analizar el dominio espectral correspondiente a las diferentes escalas de los vasos se utilizaron 5 valores para N (N=6, 8, 11, 15 y 20). Cabe mencionar que los valores de los parámetros se establecieron de manera empírica después de realizar diferentes experimentos con distintos valores de los parámetros y seleccionar los que generaban los mejores resultados.

Un vaso se detecta mejor si su dirección es ortogonal al filtro. Ya que los vasos tienen direcciones diferentes en la imagen, el análisis se lleva a cabo siguiendo 4 direcciones (líneas, columnas y dos diagonales).

La imagen original es filtrada siguiendo estos 4 barridos y se obtiene una imagen calculando los niveles máximos de niveles de gris. Para cada F_0 se lleva a cabo el mismo procedimiento y se obtienen 5 imágenes máximas.

La segunda parte de este método es la segmentación. Para cada imagen se selecciona una zona sin vasos y se analiza para obtener el nivel máximo de gris en esta zona. Este valor nos da el umbral de la imagen [3]. Utilizando este umbral se realiza la binarización de la imagen.

Posteriormente se lleva a cabo un OR lógico de estas 5 imágenes binarias para obtener la imagen final segmentada.

Por ejemplo, al aplicar el método de wavelets sobre la imagen que se muestra a continuación, se obtuvieron las imágenes binarias intermedias que se muestran posteriormente (correspondientes a cada valor de N) y finalmente una imagen que representa el OR lógico final del método.

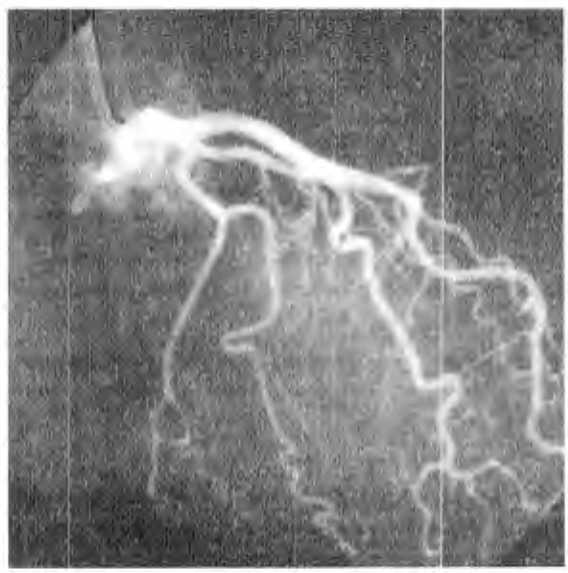


Figura 5. Imagen original

Siguiendo con la segmentación de la imagen original, se obtuvo la siguiente imagen cuando se utilizó el parámetro N=6:

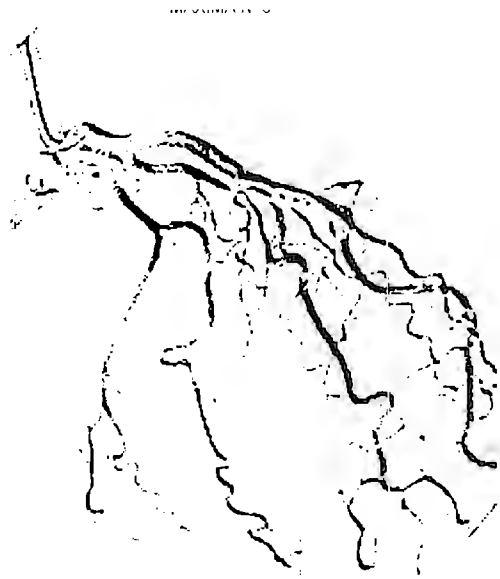


Figura 6. Filtrado utilizando N=6

Al aplicar el filtro con N=8 se obtuvo la siguiente imagen:

Figura 7. Filtrado utilizando N=8

El resultado que se obtuvo al utilizar N=11 fue el siguiente:

Figura 8. Filtrado utilizando N=11

De igual manera, al aplicar N = 15 obtuvimos:

Figura 9. Filtrado utilizando N=15

El último valor para N fue de 20 y se obtuvo la siguiente imagen:

Figura 10. Filtrado utilizando N=20

Finalmente, se realizó un OR lógico de las 5 imágenes presentadas anteriormente y se obtuvo lo siguiente:

Figura 11. Imagen final aplicando el método de wavelets

Además de aplicar el método de wavelets tal y como está definido, modificamos algunos de los parámetros con la finalidad de mejorar la imagen final:

Se realizaron filtrados con valores de N=5, 6, 7, 8, 9, 10, 12, 14, 16 y 18 y se obtuvo la siguiente imagen

Figura 12. Imagen final utilizando filtrados con valores de N=5, 6, 7, 8, 9, 10, 12, 14, 16 y 18

La implementación del algoritmo de wavelets se muestra en el Anexo C.

3.1.2 Contornos Deformables

El problema consiste en detectar un contorno de manera local a partir de una estructura deformable llamada snake.

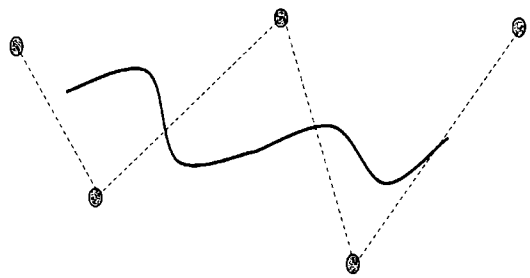


Figura 13. Puntos de control de una curva con aproximación por B-splines

Una snake está definida a partir de una serie de puntos control que describen el contorno. Para este trabajo hemos decidido usar un polinomio B-spline para generar la snake, donde cada segmento de la curva está formado y afectado solamente por 4 puntos. El objetivo es situar una snake sobre los puntos de la imagen que representan los vasos.

Hemos decidido generar la snake a partir de B-splines dado que presentan características útiles para nuestra aplicación. Algunas de ellas son:

- Controlabilidad local. Se refiere a que si afectamos un punto de control, sólo un segmento se verá modificado a la vez

- Se puede modelar una curva continua. Es posible lograr una curva continua compleja agregando más segmentos

- Pocos puntos de control. No es necesario guardar información de muchos puntos dado que cada segmento se construye con solamente cuatro

- Forman una curva suave

La interpolación por B-spline de una curva requiere de la definición de N puntos de control y la selección de un polinomio B-spline de grado n. La interpolación da como resultado la curva:

$$s(k) = \sum_{j \in Z} c(j) \bullet b^n(k-j), \qquad 0 \leq k < M \tag{3.2}$$

Donde M es el número de puntos de la snake y b^n es el polinomio B-spline de orden n.

Una vez descritas la aproximación por B-splines el método que proponemos consta de las siguientes etapas:

- Definición de una función de energía de la snake

- Selección de los puntos iniciales

- Optimización de la curva

La función de energía de la snake

La función de energía para un *snake* se define como la suma de los niveles de gris en cada punto de la snake:

$$E(s) = \sum E_i \tag{3.3}$$

Para ajustar la snake a los contornos es necesario definir para dónde se va a mover, que es lo que nosotros llamamos campo de energía externa. Es necesario realzar los contornos de la imagen. Tradicionalmente se han usado los gradientes y nosotros desarrollamos un detector de contornos de Sobel como primer posible solución al problema.

El gradiente se calculó, mediante máscaras de Sobel, en ambas direcciones, *x* y *y* respectivamente. Si después obtenemos la magnitud de ambos gradientes tenemos lo que se conoce como detector de contornos de Sobel.

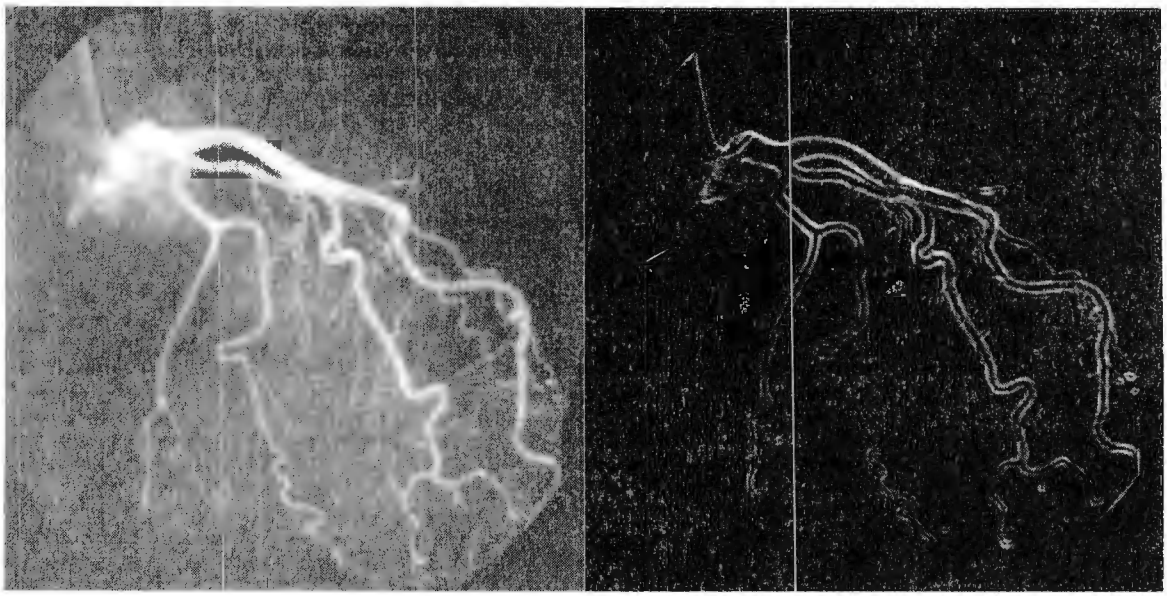


Figura 14. Imagen original (izquierda) y detector de contornos Sobel (derecha)

Existen muchos métodos para resaltar los bordes de los objetos en la imagen, pero nosotros en particular hemos decidido tomar un conjunto de filtros de Gabor por las características que exponemos a continuación.

Los filtros de Gabor pertenecen a la familia de filtros de cuadratura. Entre sus características están las siguientes:

- mediante su aplicación se puede mejorar las características de contorno
- son selectivos en orientación y frecuencia
- sus salidas pueden combinarse para producir un término robusto de energía externa para la optimización de snakes.

Los filtros de Gabor tienen un muy buen desempeño en la segmentación de contornos ya que poseen tres características importantes:

- Son funciones óptimamente localizables tanto en el espacio de frecuencia como en el de tiempo
- La forma de estas funciones se asemeja al comportamiento de algunas células simples de la retina del ojo
- Son filtros pasobandas que son capaces de extraer frecuencias específicas de una imagen

Los filtros de Gabor pueden ser de dos dimensiones y se pueden diseñar para que detecten contornos en cualquier dirección, de una manera muy precisa. Las salidas que se obtienen con los filtros de Gabor son muy apropiadas para identificar características vasculares y para utilizar en la optimización de snakes [4].

El filtro de Gabor de una dimensión está definido por las siguientes funciones:

$$S(x) = e^{\left(-\frac{x^2}{2\sigma^2}\right)} \cdot \sin(2\pi v_O x) \tag{3.4}$$

$$C(x) = e^{\left(-\frac{x^2}{2\sigma^2}\right)} \cdot \cos(2\pi v_O x) \tag{3.5}$$

en donde x es la distancia en el eje x, σ determina el ancho de la envolvente gaussiana y v_0 es la frecuencia de modulación. S(x) define a la función impar asimétrica y C(x) define a la función par simétrica de Gabor.

El filtro de Gabor de una dimensión se puede extender a dos dimensiones si se multiplican las funciones de Gabor de una dimensión por un filtro gaussiano pasobajas en la dirección ortogonal [4].

Existe otra versión de los filtros de Gabor denominada filtros S-Gabor ("Stretched-Gabor") que busca eliminar la respuesta de DC diferente de cero que se presenta en la función par del filtro de Gabor. Esto se realiza incluyendo un barrido a bajas frecuencias en las funciones seno y coseno. Las funciones del filtro S-Gabor están definidas como:

$$S(x) = e^{\left(-\frac{x^2}{2\sigma^2}\right)} \cdot \sin(2\pi v_O x \psi(x)) \tag{3.6}$$

$$C(x) = e^{\left(-\frac{x^2}{2\sigma^2}\right)} \cdot \cos(2\pi v_O x \psi(x)) \tag{3.7}$$

$$\psi(x) = k \cdot e^{\left(-\lambda\left(\frac{x}{\sigma}\right)^2\right)} \tag{3.8}$$

en donde k es la contribución relativa del barrido frecuencias y λ determina la velocidad del barrido frecuencial.

Para valores dados de k, σ y v_0, con (σ · v0)>0.38 existe al menos un valor de λ que satisface la condición de cero DC, estos es, que la integral dentro del intervalo que dura el filtro par es cero.

La función ψ(x) produce un aumento en la longitud de onda de las funciones seno y coseno conforme aumenta la distancia desde el origen, agregando más peso al lado negativo del filtro par. Esto agrega suficiente área negativa para hacer posible una solución de la integral, igual a cero.

Los filtros de Gabor de 1-D se pueden extender a dos dimensiones mediante la multiplicación de las máscaras en 1-D por un filtro gaussiano en la dirección 'y'. El efecto de la envolvente gaussiana orientada perpendicularmente al eje principal del filtro de Gabor es el de promediar la información de la intensidad a lo largo de los detalles de los contornos o líneas que se desean mejorar. En este caso, la orientación del filtro se realiza mediante la utilización de una matriz de rotación de un ángulo apropiado al filtro de 2-D

$$S(x) = e^{\left(-\frac{1}{2}\left(\frac{x^2}{\sigma^2}+\frac{y^2}{\sigma^2}\right)\right)} \cdot \sin(2\pi v_O x \psi(x)) \tag{3.9}$$

$$C(x) = e^{\left(-\frac{1}{2}\left(\frac{x^2}{\sigma^2}+\frac{y^2}{\sigma^2}\right)\right)} \cdot \cos(2\pi v_O x \psi(x)) \tag{3.10}$$

$$\psi(x) = k \cdot e^{\left(-\lambda\left(\frac{x}{\sigma}\right)^2\right)} \tag{3.11}$$

La elección de la anchura de las envolventes gaussianas, σx y σy, determinan la frecuencia y la selectividad de la orientación del filtro S-Gabor de 2-D, respectivamente. σy controla el tiempo y la continuidad, así como qué tan cerca del eje principal del filtro deben estar las características de la imagen para poder ser detectadas. σx controla qué tan abruptamente debe de sobresalir el contorno o la línea, comparado con el fondo, para poder ser detectado [4].

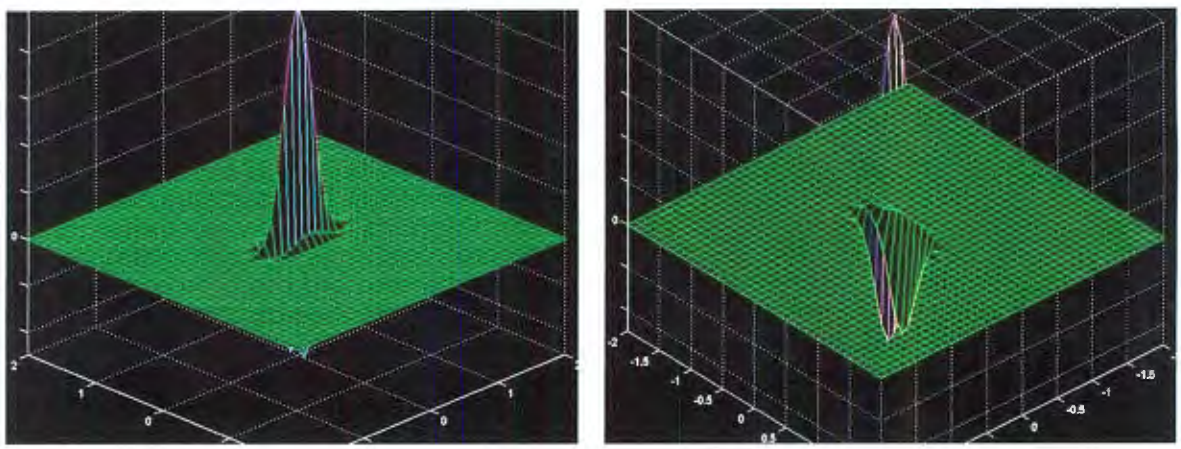


Figura 15. Función impar C(x) del filtro de S-Gabor 2D

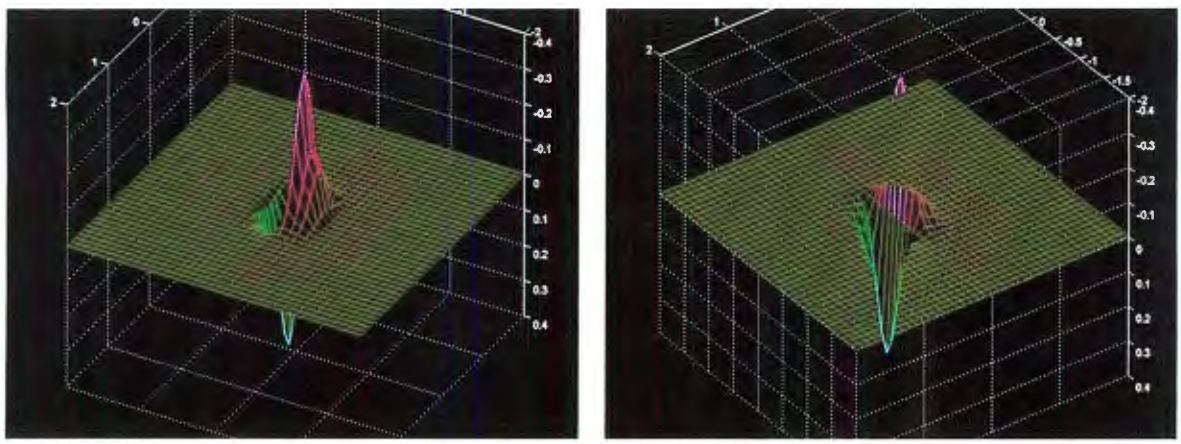


Figura 16. Función par S(x) del filtro de S-Gabor 2D

La magnitud y la dirección de las variaciones en la intensidad de la imagen describen contornos contenidos en una imagen. La intensidad de la imagen filtrada con filtros S-Gabor en cada punto puede utilizarse para conducir a una snake que minimice energías a los contornos presentes en la imagen. El campo de energía, *E,* está definido como la magnitud vectorial de las salidas del filtro obtenida mediante la convolución de los filtros S-Gabor par e impar de orientación θ con la imagen I.

$$E(\theta) = \sqrt{[C(\theta) * I]^2 + [S(\theta) * I]^2} \qquad (3.12)$$

Cabe mencionar que la ubicación de picos de energía corresponde exactamente a la ubicación de estas características (contornos). Sin embargo, la energía no contiene

ningún tipo se información acerca de la forma específica que generó la respuesta. Sin embargo, la fase sí contiene información sobre la característica de la forma que generó la respuesta, ya sea una línea o un contorno.

$$\phi(\theta) = \tan^{-1}\left[\frac{S(\theta) * I}{C(\theta) * I}\right] \tag{3.13}$$

Para crear un campo de energía de características par o impar de cualquier orientación, se puede realizar una convolución de un banco de filtros de diferentes orientaciones y se calcula la energía para cada orientación. La respuesta de máxima energía a través del banco de filtros en cada punto de la imagen define un campo de energía compuesto:

$$\varepsilon(x,y) = \max_{\theta} E(x,y,\theta) \tag{3.14}$$

Este tipo de filtrado se ha desarrollado para utilizarse como etapa inicial del método de snakes.

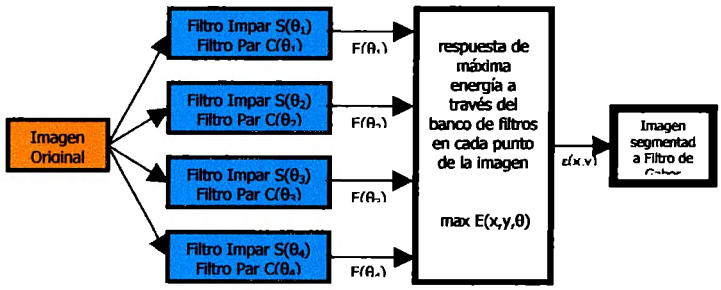


Figura 17. Implementación del filtro de Gabor

Los parámetros que se seleccionaron para definir las funciones del filtro S-Gabor implementado fueron los siguientes:

Para la función impar:
$\sigma x = 1.5$
$k = 0.9$
$v_0 = 0.39$
$\sigma y = 2$
$\lambda = -0.4$

Para la función impar:
$\sigma x = 0.9$
$k = 0.9$
$v_0 = 0.39$
$\sigma y = 2$
$\lambda = -0.4$

Todos los valores utilizados se obtuvieron de manera experimental al aplicar el filtrado con diferentes valores para cada parámetro y finalmente se escogieron los parámetros que nos generaban los mejores resultados para nuestros requerimientos.

El código utilizado para implementar el algoritmo de filtros de Gabor se muestra en el Anexo D.

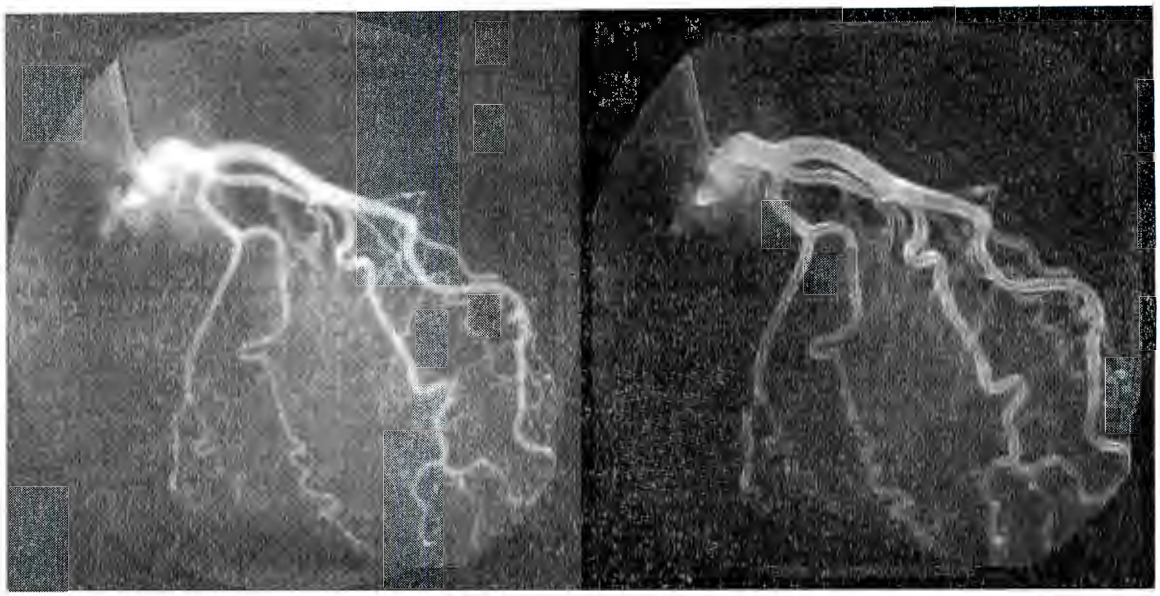


Figura 18. Original (izquierda) y filtrada por Gabor (derecha)

Por medio de la información arrojada por los filtros de Gabor obtenemos una imagen que nos servirá para ubicar a un conjunto de snakes que nos representan los vasos.

Selección de los puntos iniciales

Para generar la snake es necesario que definamos un conjunto de puntos de control iniciales. El método es especialmente sensible a la ubicación de estos puntos iniciales. Por el momento nosotros seleccionamos de manera manual los puntos de control iniciales para cada snake.

Para el análisis presentado los puntos de control en los extremos los dejamos fijos para controlar que la spline no se alejara demasiado de la región de interés aunque el algoritmo permite que todos los puntos de control puedan moverse. Aprovechamos también el fijar esos puntos para reducir el número de iteraciones necesarias para converger a la solución óptima porque se reducen los grados de libertad del sistema.

Optimización de la curva

- Se inicializan los puntos de control
- Se mueve un sólo punto de control dentro de una matriz de 5x5 y se fija donde se encontró la energía mínima
- Se repite el proceso con los siguientes puntos de control para terminar así la primera iteración

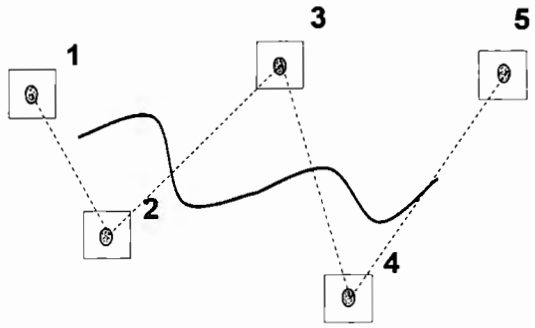


Figura 19. Los recuadros representan la matriz 5x5 en la que se mueve el punto de control

Las iteraciones se detienen cuando:

1. La snake anterior no difiere significativamente de la recién generada
2. Fijamos un número determinado de iteraciones

3.1.3 Momentos Geométricos

Las técnicas de detección de ejes han sido muy investigadas en la literatura. Hay muchos operadores que son usados ampliamente en la actualidad como los de Roberts, Sobel, Prewitt y otros. Sin embargo los operadores basados en la estimación de momentos tienen un buen acotamiento y son muy robustos contra el ruido según estudios de Shu, Bao y Zhan [15].

Utilizamos este método para segmentar una imagen angiográfica, para poder detectar los contornos de las arterias coronarias.

Este método se basa en la diferencia de color (en escala de grises) que existe entre dos pixeles. Si la diferencia es grande de un pixel a otro, sabemos que tenemos un contorno.

Supongamos que tenemos el nivel de gris sobre una superficie *f(x)* y el valor del nivel representa una altura, si hacemos un corte transversal de dicha superficie veríamos la figura siguiente.

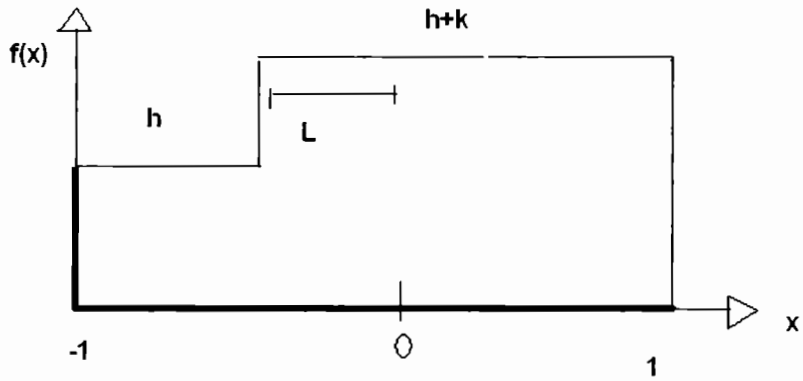


Figura 20. Modelo ideal del contorno en una dimensión

En donde 0 representa la posición del pixel que se está analizando, *L* representa la distancia desde el pixel hasta el contorno, *h* una altura representada por el nivel de gris en la superficie, *h+k* el nivel de gris de la segunda superficie y *k* la diferencia de niveles de gris entre las dos superficies. Por lo tanto *x* sería la posición y *F(x)* representa al nivel de gris en función de la posición.

El momento p de una función de la posición f(x) es:

$$Mp = \int x^p f(x)dx \tag{3.15}$$

Por lo que los momentos 0, 1 y 2 quedan de la siguiente forma:

$$M_0 = h\int_{-1}^{1} dx + k\int_{l}^{1} dx = 2h + k(1-l) \tag{3.16}$$

$$M_1 = h\int_{1}^{1} xdx + k\int_{l}^{1} xdx = \tfrac{1}{2}k(1-l^2) \tag{3.17}$$

$$M_2 = h\int_{1}^{1} x^2dx + k\int_{l}^{1} x^2dx = \tfrac{2}{3}h + \tfrac{1}{3}k(1-l^3) \tag{3.18}$$

Podemos apreciar que nos resulta un sistema de 3 ecuaciones con 3 incógnitas, de manera que podemos despejar:

$$l = \frac{3M_2 - M_0}{2M_1} \tag{3.19}$$

$$k = \frac{2M_1}{(1-l^2)} \tag{3.20}$$

$$h = \frac{1}{2}\left[M_0 - k(1-l)\right] \tag{3.21}$$

Si analizamos de la misma manera la imagen en 2 dimensiones y se cambia uno de los sistemas coordenados cartesianos por cilíndricos se obtiene un corte como el de la figura mostrada.

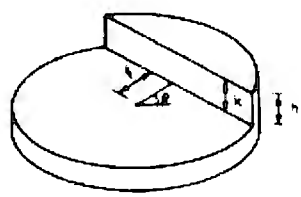


Figura 21. Modelo de contorno ideal en dos dimensiones caracterizado por *h, k, θ* y *L*

Donde θ será el ángulo desde el pixel hasta el contorno, k la diferencia de nivel de grises entre la superficie del pixel y la superficie del contorno, L la distancia del pixel al contorno y h el nivel de gris del contorno.

De esta forma podemos calcular el momento p, q en dos dimensiones con esta fórmula:

$$M_{pq} = \sum_{x} \sum_{y} \iint_{A_{xy}} x^{p} y^{q} f(x,y)\,dx\,dy \tag{3.22}$$

Haciendo una rotación la ecuación nos queda:

$$M'_{pq} = \sum_{r=0}^{p} \sum_{s=0}^{q} \binom{p}{r} \binom{q}{s} (-1)^{q-s} (\cos\phi)^{p-r+s} * (\sin\phi)^{q+r-s} M_{p+q-r-s,\,r+s} \tag{3.23}$$

Calculando para las primeras dos combinaciones de momentos (00,01,10,20,02,11) los parámetros buscados son:

$$\theta = \tan^{-1} \frac{M_{01}}{M_{10}} \tag{3.24}$$

$$k = \frac{3M'_{10}}{2(1-l^2)^3} \tag{3.25}$$

$$h = \frac{1}{2\pi}\left[2M'_{00} - k\left(\pi - 2\sin^{-1} l - 2l\sqrt{1-l^2}\right)\right] \tag{3.26}$$

$$l = \frac{4M'_{20} - M'_{00}}{3M'_{10}} \tag{3.27}$$

Con estos parámetros h, L, θ y k generamos una nueva imagen que se puede segmentar con mucha mayor precisión que los métodos convencionales.

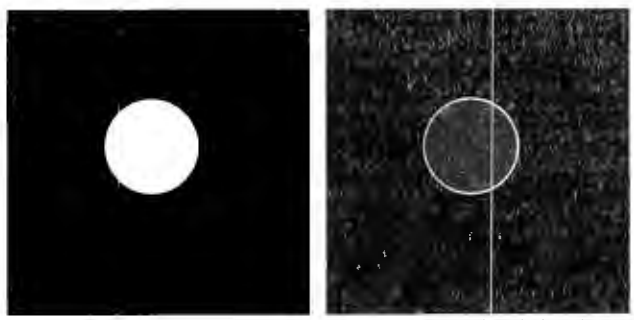


Figura 22. Imagen de prueba original (izquierda) e imagen segmentada por aproximación lineal (derecha)

Aproximación Parabólica

El método antes descrito se basa en una aproximación al contorno de tipo lineal en donde se trata de aproximarse al contorno con pequeñas líneas rectas. Sin embargo, en contornos curvos la mejor aproximación no es esta, por lo que se implementó el algoritmo con una aproximación parabólica. Ésta al igual que la lineal se basa en la diferencia en la escala de grises entre los pixeles y también en los momentos para su implementación.

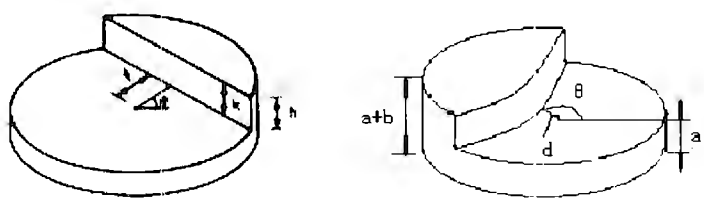


Figura 23. a) Aproximación Lineal b) Aproximación Parabólica

El desarrollo matemático para explicar la aproximación parabólica es el siguiente:
Ecuación de la parábola:

$$y^2 = kx - h \tag{3.28}$$

Ecuación del círculo:

$$x^2 + y^2 = r^2 \tag{3.29}$$

Ahora consideremos un punto A dentro de la figura y que dicho punto es el de intersección entre el círculo y la parábola.

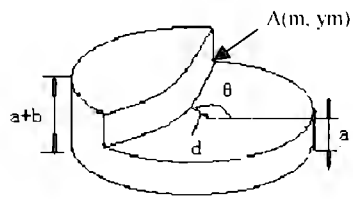


Figura 24. Visualización gráfica de los parámetros

Si dicho punto tiene coordenadas A(m, ym) el despeje para los parámetros k y h sería el siguiente:

$$k = \frac{r^2 - m^2}{m - d} \tag{3.30}$$

$$h = \frac{r^2 - m^2}{m - d} \times d \tag{3.31}$$

El parámetro *d* a partir de la ecuación de momentos se puede deducir como:

$$d = \frac{7M'_{20} + 5M'_{02} - 3r^2 M'_{00}}{4M'_{10}} \tag{3.32}$$

La ecuación para *b* se resulta de la siguiente forma:

$$b = \frac{15M'_{10}}{2\ r^2 - m^2(m^2 - 2md - 4d^2 + 5r^2)} \tag{3.33}$$

Dejemos que:

$$M = \frac{r^2 M'_{00} - 4M'_{02}}{M'_{10}} \tag{3.34}$$

Entonces

$$3m^3 - (8d + M)m^2 + (7r^2 + 2Md)m + (4Md^2 - 2dr^2 - 5Mr^2) = 0 \tag{3.35}$$

Resolviendo la ecuación de 3er orden se puede obtener el valor de *m* que pertenece al rango $0 \leq m < r$.

Para calcular el parámetro *b* a partir de los momentos sólo nos hacía falta el cálculo de *m* por lo que ahora ya se puede obtener el parámetro *b* que es la diferencia de niveles de gris de los contornos.

Algoritmo
El algoritmo computacional que se utilizó para realizar la segmentación por momentos geométricos lleva los siguientes pasos:

- Cálculo de los momentos
- Cálculo de los momentos primados (las rotaciones)
- Cálculo de los parámetros
- Resultado final

Cálculo de los Momentos. Computacionalmente, esta parte del proceso se lleva a cabo mediante la utilización de un algoritmo que convoluciona la imagen original con las máscaras propias de cada momento. Las máscaras son las siguientes:

0.0219	0.1231	0.1573	0.1231	0.0219
0.1231	0.1600	0.1600	0.1600	0.1231
0.1573	0.1600	0.1600	0.1600	0.1573
0.1231	0.1600	0.1600	0.1600	0.1231
0.0219	0.1231	0.1573	0.1231	0.0219

Mascara M00

-0.0098	-0.0352	0.0000	0.0352	0.0098
-0.0352	-0.0256	0.0000	0.0256	0.0352
0.0000	0.0000	0.0000	0.0000	0.0000
0.0352	0.0256	0.0000	-0.0256	-0.0352
0.0098	0.0352	0.0000	-0.0352	-0.0098

Mascara M11

-0.0147	-0.0469	0.0000	0.0469	0.0147
-0.0933	-0.0640	0.0000	0.0640	0.0933
-0.1253	-0.0640	0.0000	0.0640	0.1253
-0.0933	-0.0640	0.0000	0.0640	0.0933
-0.0147	-0.0469	0.0000	0.0469	0.0147

Mascara M10

0.0147	0.0933	0.1253	0.0933	0.0147
0.0469	0.0640	0.0640	0.0640	0.0469
0.0000	0.0000	0.0000	0.0000	0.0000
-0.0469	-0.0640	-0.0640	-0.0640	-0.0469
-0.0147	-0.0933	-0.1253	-0.0933	-0.0147

Mascara M01

0.0099	0.0194	0.0021	0.0194	0.0099
0.0719	0.0277	0.0021	0.0277	0.0719
0.1019	0.0277	0.0021	0.0277	0.1019
0.0719	0.0277	0.0021	0.0277	0.0719
0.0099	0.0194	0.0021	0.0194	0.0099

Mascara M20

0.0099	0.0719	0.1019	0.0719	0.0099
0.0194	0.0277	0.0277	0.0277	0.0194
0.0021	0.0021	0.0021	0.0021	0.0021
0.0194	0.0277	0.0277	0.0277	0.0194
0.0099	0.0719	0.1019	0.0719	0.0099

Mascara M02

Cálculo de los momentos primados (las rotaciones). Mediante operaciones matemáticas se calculan las rotaciones o momentos primados siguientes:

$$M'_{00} = M_{00} \tag{3.36}$$

$$M'_{10} = M_{01}^2 + M_{10}^2 \tag{3.37}$$

$$M'_{20} = \frac{M_{10}^2 M_{20} + 2M_{01}M_{10}M_{11} + M_{01}^2 M_{02}}{M_{01}^2 + M_{10}^2} \tag{3.38}$$

$$M'_{02} = \frac{M_{10}^2 M_{02} - 2M_{01}M_{10}M_{11} + M_{01}^2 M_{20}}{M_{01}^2 + M_{10}^2} \tag{3.39}$$

Cálculo de los parámetros. Una vez teniendo todos los momentos necesarios, se procede al cálculo de los parámetros necesarios y los auxiliares matemáticos (m, M) para cada aproximación. En el caso de la parabólica son:

$$d = \frac{7M'_{20} + 5M'_{02} - 3r^2 M'_{00}}{4M'_{10}} \tag{3.40}$$

$$M = \frac{r^2 M'_{00} - 4M'_{02}}{M'_{10}} \tag{3.41}$$

$$3m^3 - (8d + M)m^2 + (7r^2 + 2Md)m + (4Md^2 - 2dr^2 - 5Mr^2) = 0 \tag{3.42}$$

$$b = \frac{15M'_{10}}{2\ \ r^2 - m^2(m^2 - 2md - 4d^2 + 5r^2)} \tag{3.43}$$

Resultado final. Una vez obtenidos todos los parámetros necesarios se obtiene la imagen final del todo el algoritmo.

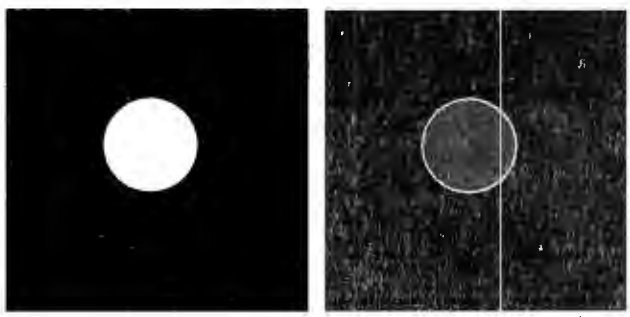


Figura 25. Imagen de prueba original (izquierda) e imagen segmentada por aproximación parabólica (derecha)

3.2 Validación de Métodos

Para la etapa de pruebas de los algoritmos implementados se crearon dos imágenes básicas: una recta y una curva.

Recta. Esta imagen se construyó para simular una arteria recta de 20 pixeles de diámetro. Primeramente se creó un rectángulo de 280 x 20 pixeles. Posteriormente, se construyó una matriz de 512 x 512 pixeles y a cada uno de esos pixeles se le asignó un valor de 0 (equivalente a un nivel de color negro). La razón por la que se construyó esta matriz fue que todas las imágenes angiográficas que se han utilizado en este trabajo presentan dichas dimensiones.

Después de esto, se utilizó una función circular para asignarle niveles de gris a cada uno de los pixeles encerrados dentro del rectángulo creado. Cabe mencionar que los niveles de gris de forma circular se asignaron considerando el diámetro del círculo de forma vertical dentro del rectángulo, es decir, con un racio de 10 pixeles. Con esto se asegura que la arteria recta simulada presente características de color (niveles de gris) muy parecidas a las de una arteria real. Algo muy importante dentro de esta última característica es que el valor del nivel de gris asignado a cada pixel de la arteria simulada se manipuló de tal manera que el máximo nivel de gris presente en la parte más blanca de la arteria fuera de 200.

La arteria simulada resultante fue la siguiente:

Figura 26. Arteria simulada en forma recta

El código utilizado para la generación de esta imagen se muestra en el anexo A.

Curva. La construcción de esta figura fue mucho más complicada que la recta descrita anteriormente Para esta arteria también se eligió un diámetro de 20 pixeles. Primeramente, se definió una función exponencial, la cual constituiría el eje de nuestra arteria simulada.

Después, con la finalidad de obtener las curvas que conformarían los contornos de la figura y las cuales debían seguir la misma forma que la curva central definida con la función exponencial, se calculó la pendiente de la tangente en cada punto de la curva central y la pendiente que debería tener una recta perpendicular a dicha tangente. Con este valor de la pendiente se calculó el ángulo de la recta perpendicular. Ya conociendo

los valores de la distancia a la que debería estar cada una de las curvas de contorno, conociendo las coordenadas de cada uno de los puntos de la curva central y conociendo el ángulo de la perpendicular a la tangente, se calcularon los puntos correspondientes en cada una de las curvas de contorno.

Enseguida, se utilizó una rutina recursiva de interpolación lineal para determinar, a partir de los puntos inicial y final, los puntos que conformaban cada una de las rectas ortogonales. Una vez hecho esto, se procedió a asignar a cada punto de cada recta ortogonal un nivel de gris siguiendo igualmente una función circular para calcular el nivel de gris correspondiente, de tal manera que el nivel más blanco quedara en el centro de la figura (sobre la curva central).

La primera parte del proceso de generación de esta arteria simulada nos dio como resultado la siguiente figura:

Figura 27. Primer paso en la generación de la arteria simulada en forma curva

Una vez que se agregaron niveles de gris a cada uno de los puntos dentro de la figura curva, se colocó la misma en una posición de la matriz de 512 x 512 pixeles y finalmente la arteria simulada fue la siguiente:

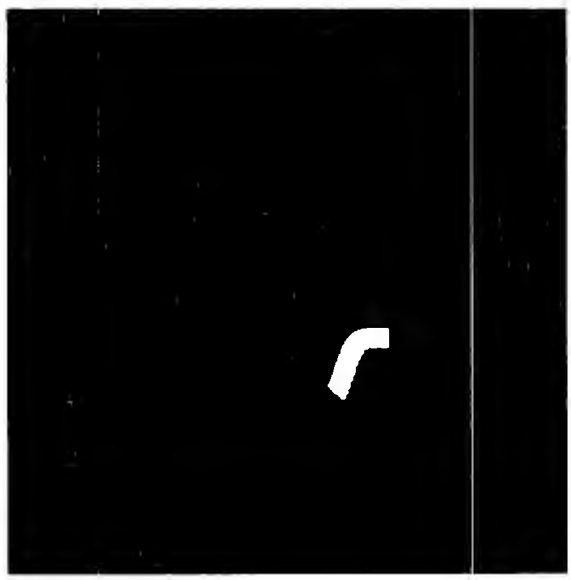


Figura 28. Aspecto final de la arteria simulada en forma curva

El código utilizado para generar esta imagen se encuentra en el anexo B.

A continuación se presentan las imágenes obtenidas al procesar las figuras de arterias simuladas (tanto la recta como la curva) con los distintos algoritmos:

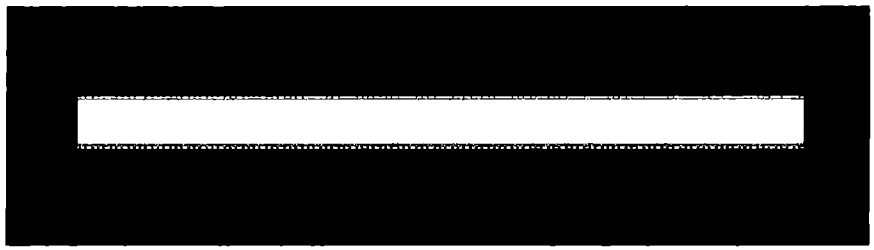


Figura 29. Imagen original de arteria simulada (recta)

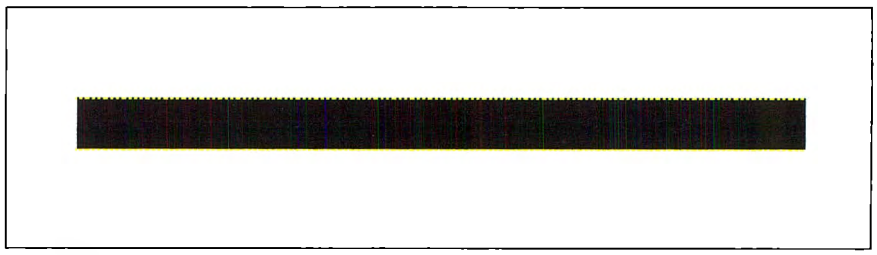


Figura 30. Imagen final al aplicar método de wavelets

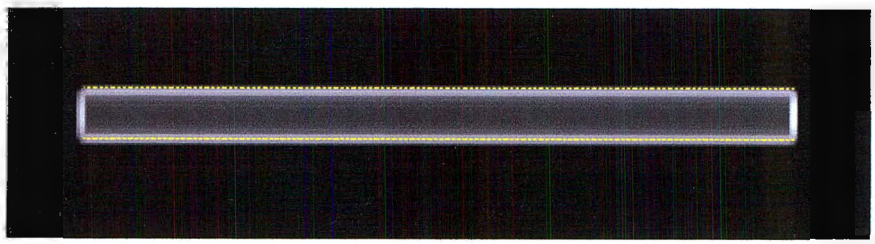


Figura 31. Imagen final al aplicar método de momentos

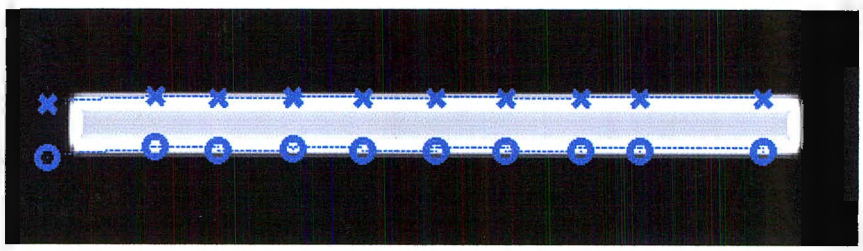


Figura 32. Imagen final al aplicar método de splines

Figura 33. Imagen original de arteria simulada (curva)

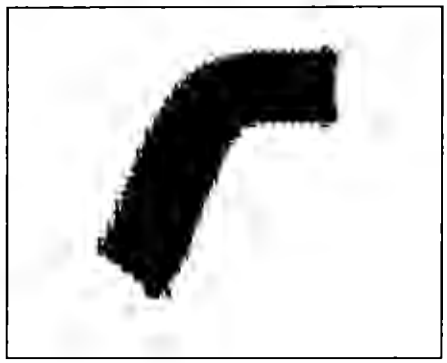


Figura 34. Imagen final al aplicar método de wavelets

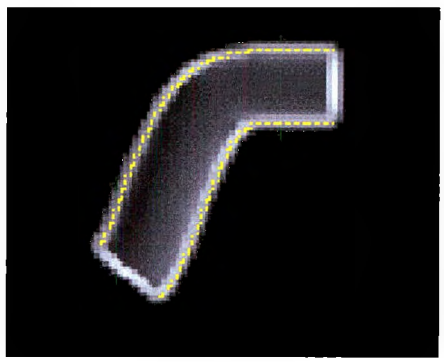
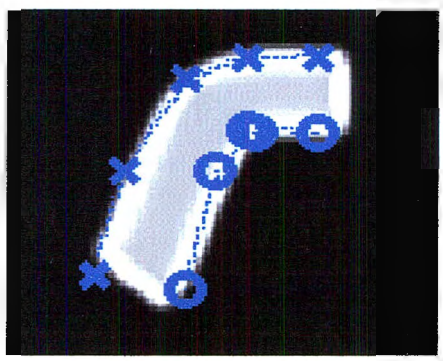


Figura 35. Imagen final al aplicar método de momentos geométricos



3.3 Herramientas Computacionales de Ayuda al Diagnóstico

La caracterización y detección de patologías de manera automática es de suma importancia para el mejoramiento del diagnóstico y cirugía en sistemas de cirugía asistida por computadora. En esta parte propondremos una metodología para la caracterización de estenosis mediante la estimación de los diámetros mínimo y máximo en una zona local de una arteria. Las estenosis se caracterizan de acuerdo al grado de estrechamiento en la arteria. En esta propuesta de caracterización partimos de imágenes de proyección, es decir, que la estenosis es aparente y el grado de estrechamiento depende de la proyección. Este método permite al menos encontrar la mejor proyección para caracterizar la estenosis. La metodología propuesta tiene las siguientes etapas:

- Detección de los bordes de una zona de una arteria mediante algoritmos de detección
- Estimación de los diámetros aparentes en la zona de estrechamiento y en la zona sin estrechamiento
- Cuantificación de la severidad de la estenosis

3.3.1 Detección de Bordes

La detección de bordes se realiza mediante el algoritmo de contornos deformables con B-splines. Este algoritmo presenta una naturaleza local que permite escoger la zona de la arteria donde se va a realizar la caracterización en contraposición con otros algoritmos que tienen un carácter global y en cuyo caso sería necesario agregar etapas adicionales para localizar la zona de interés (wavelets, momentos geométricos, etc.).

3.3.2 Estimación de Diámetros Aparentes

Una vez localizada la zona de interés mediante dos snakes correspondientes a cada borde de la arteria se necesita localizar la estenosis y cuantificarla. Los métodos tradicionales [4] [6] construyen o detectan la línea central para estimar los diámetros. Nosotros proponemos un método de correspondencia de bordes utilizado para la estructuración de contornos [13] para la estimación de los diámetros.

Este método de correspondencia se basa en la comparación de dos cadenas de caracteres [12], en este caso representadas por los dos bordes de la arteria. El algoritmo da la correspondencia entre los pixeles de cada contorno cuyas curvaturas son más parecidas. Gráficamente se representan mediante líneas rectas. Este conjunto de líneas rectas se conoce como ***Traza***. Estas líneas rectas son perpendiculares a los

contornos y pueden ser utilizadas para la estimación de los diámetros, como se ve en la figura.

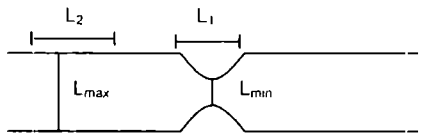


Figura 37. Simulación de una arteria con estenosis

Estimación de diámetro mínimo (Lr)
El diámetro mínimo corresponde a la recta de longitud mínima de la traza. Se toma una vecindad definida por L1 (Figura 37) para estimar el diámetro mínimo.

Estimación de diámetro máximo (Lm)
El diámetro mínimo corresponde a la recta de longitud mínima de la traza. Se toma una vecindad definida por L2 (Figura 37) para estimar el diámetro máximo.

3.3.3 Cuantificación de Severidad de la Estenosis

La severidad de estenosis se calcula mediante el porcentaje de diámetro obstruido (Ls) y mediante el porcentaje de área obstruida (As), definidas como sigue:

$$L_s = \%Length\ stenosis = \left(1 - \frac{L_r}{L_m}\right) \times 100 \tag{3.44}$$

$$A_s = \%Area\ stenosis = \left(1 - \frac{A_r}{A_m}\right) \times 100 \tag{3.45}$$

Lr corresponde al diámetro mínimo y Lm al diámetro máximo. Suponiendo un área transversal circular, Ar corresponde al área en la zona de estrechamiento y Am el área en la zona de diámetro máximo.

4 Resultados

4.1 Segmentación Utilizando el Algoritmo de Wavelets

El método de wavelets se aplicó sobre cuatro imágenes de una secuencia formada por 15 imágenes (secuencia SEG2). Las imágenes obtenidas fueron las siguientes:

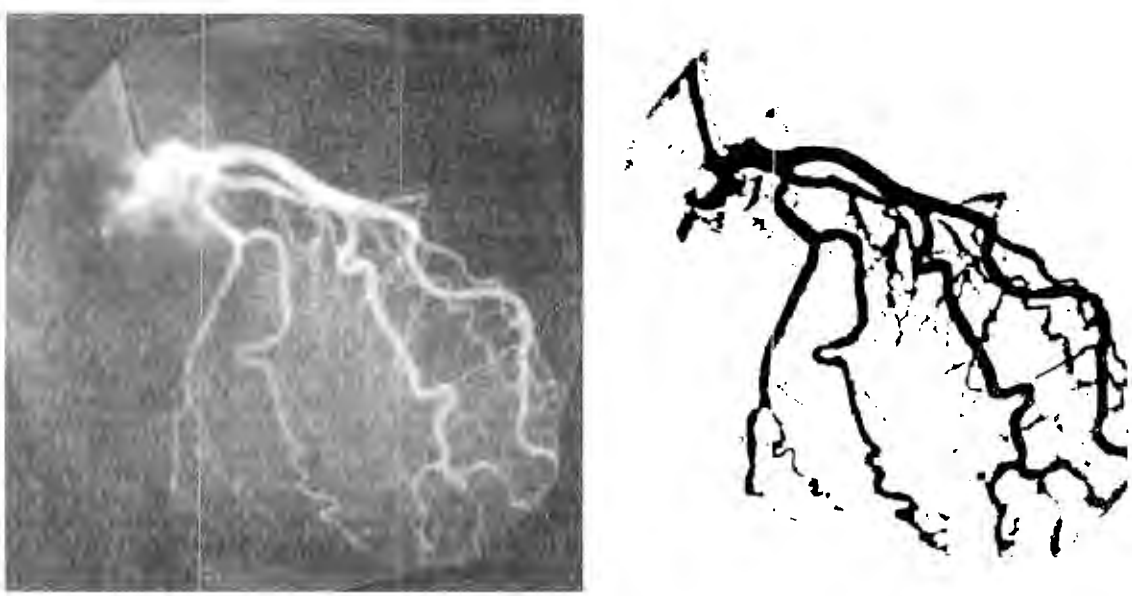


Figura 38. Imagen original (izquierda) y segmentada por wavelets (derecha)

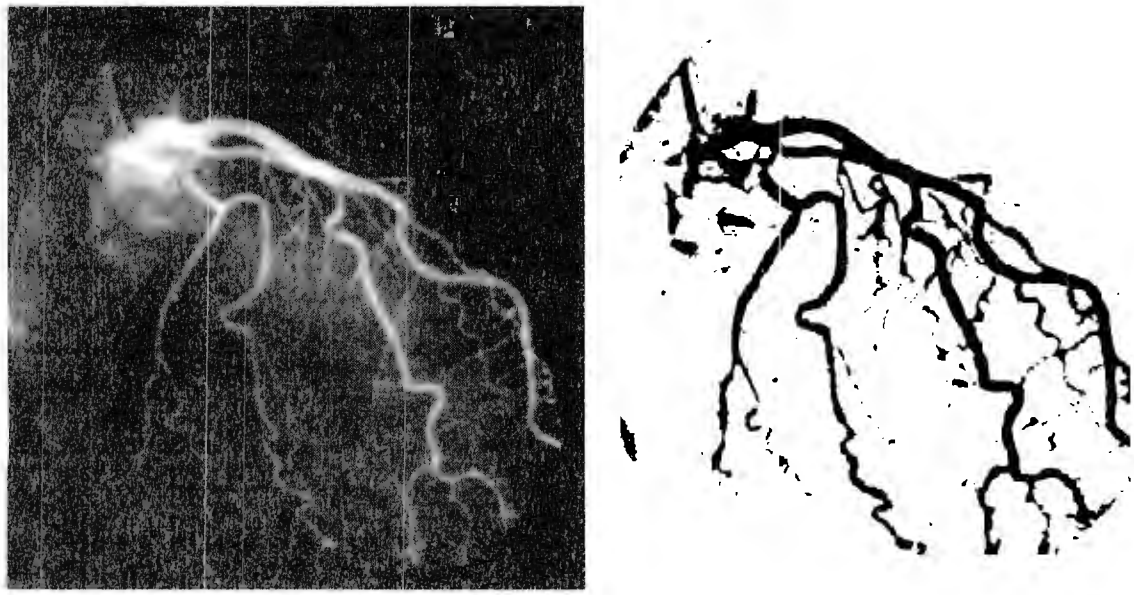


Figura 39. Imagen original (izquierda) y segmentada por wavelets (derecha)

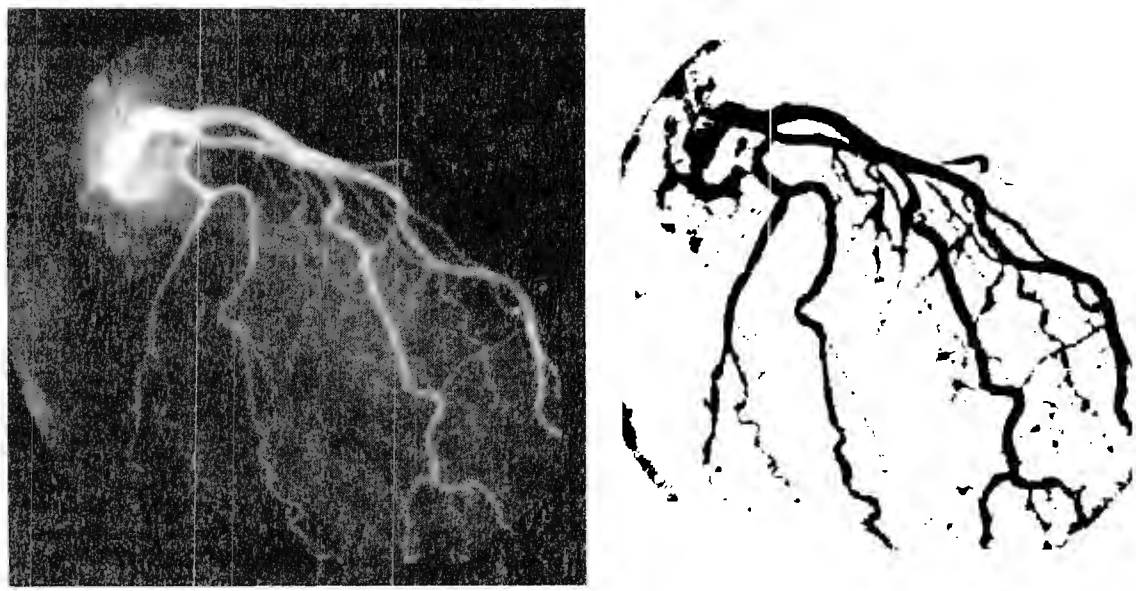


Figura 40. Imagen original (izquierda) y segmentada por wavelets (derecha)

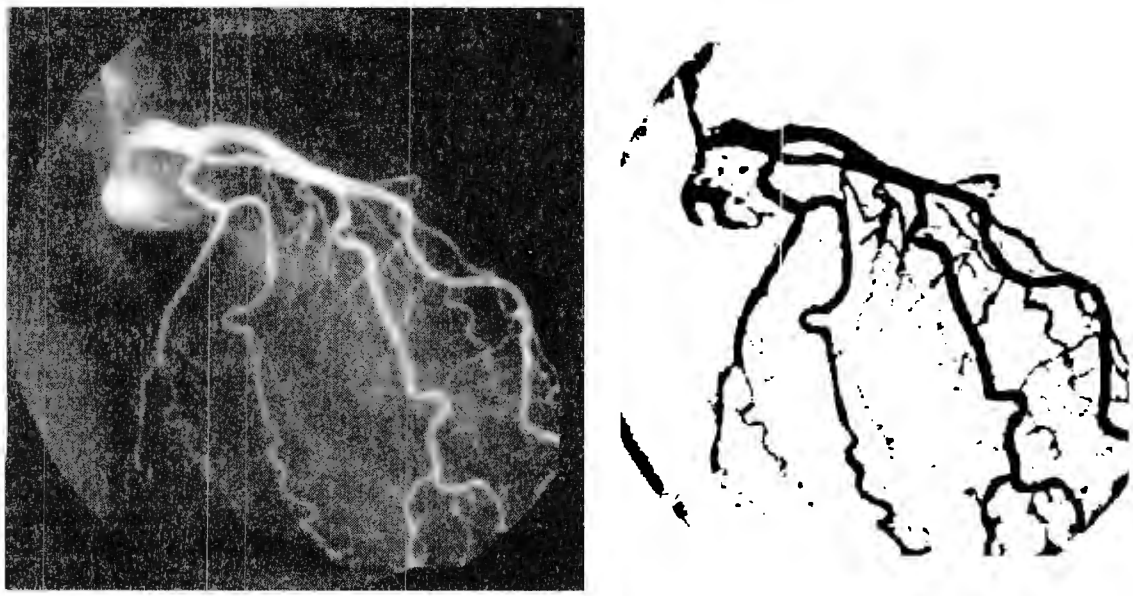


Figura 41. Imagen original (izquierda) y segmentada por wavelets (derecha)

Al comparar cada una de las imágenes originales con las procesadas con el método de wavelets se puede observar que las imágenes segmentadas son muy precisas, pues detectan las principales estructuras vasculares en las imágenes originales.

Sin embargo, también se observan algunos detalles que en la imagen original aparecen con niveles de gris similares a los de las arterias, por lo que también aparecen en la imagen segmentada. También, las imágenes segmentadas fallan en mostrar las arterias muy delgadas, ya que éstas se ven, en el mejor de los casos, como puntos o manchas y no como tubos continuos como aparecen las arterias más gruesas.

4.2 Segmentación Utilizando el Algoritmo de Momentos Geométricos de una Imagen

Se muestran a continuación los resultados para 4 imágenes angiográficas distintas utilizando la aproximación parabólica.

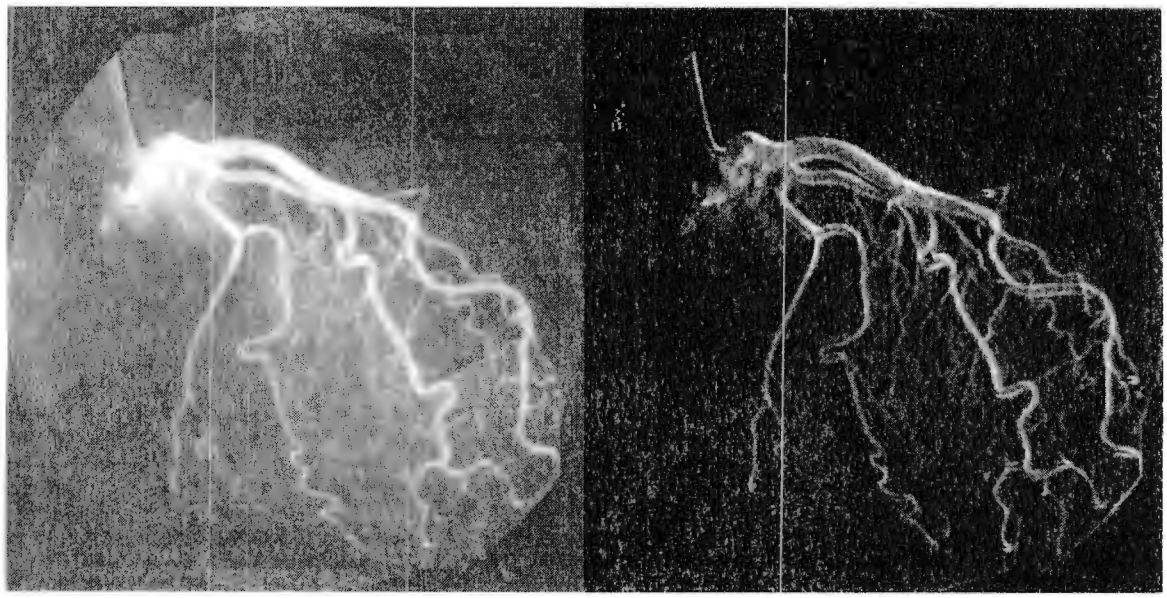


Figura 42. Imagen original (izquierda) y segmentada por momentos de aproximación parabólica

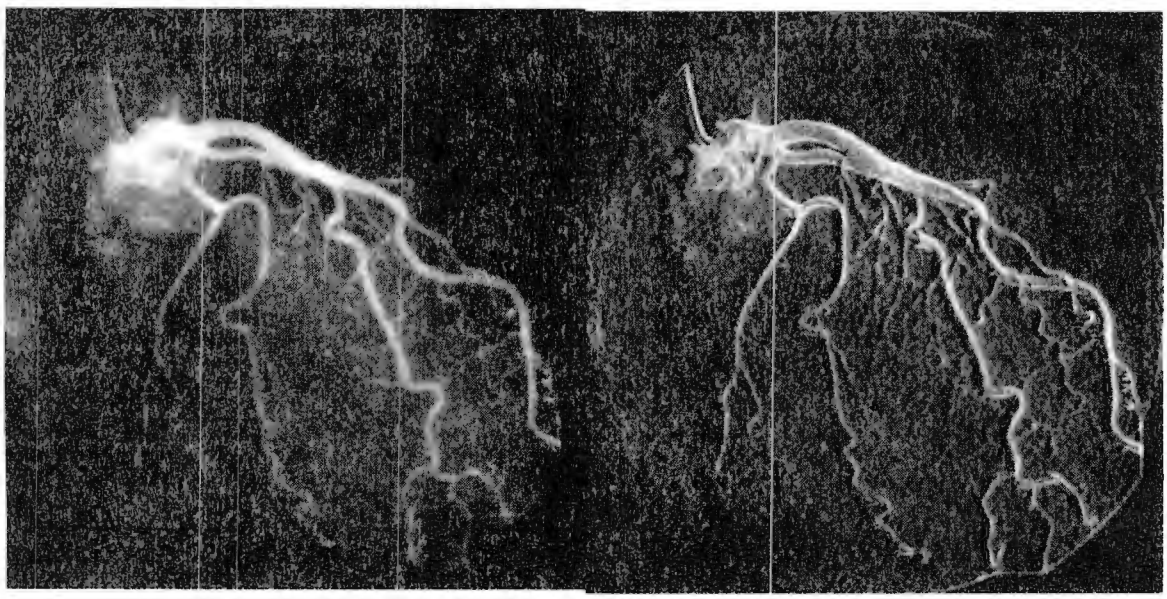


Figura 43. Imagen original (izquierda) y segmentada por momentos de aproximación parabólica

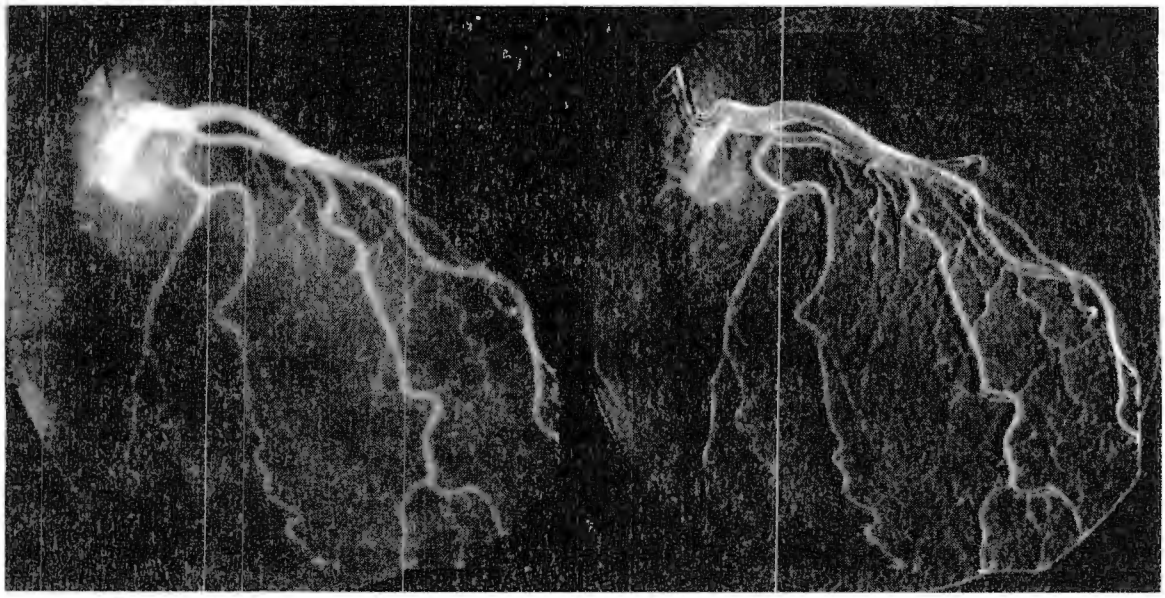


Figura 44. Imagen original (izquierda) y segmentada por momentos de aproximación parabólica

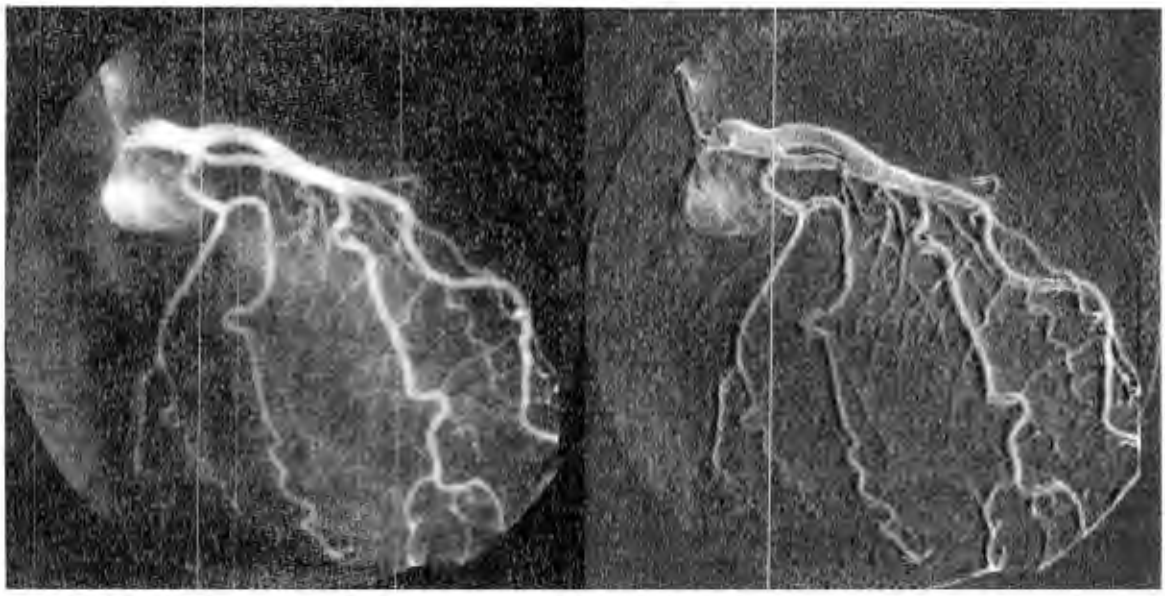


Figura 45. Imagen original (izquierda) y segmentada por momentos de aproximación parabólica

En las gráficas anteriores se puede ver como el método de momentos geométricos por aproximación parabólica es efectivo para la segmentación y funciona para diferentes imágenes por lo que se puede considerar un método robusto en este sentido.

4.3 Segmentación Utilizando Algoritmo de Deformación mediante B-Splines

El método de B-splines requiere de dos condiciones para arrancar:

1. Obtener un campo de energía externa que obtenemos con el filtrado S-Gabor
2. Selección de los puntos iniciales

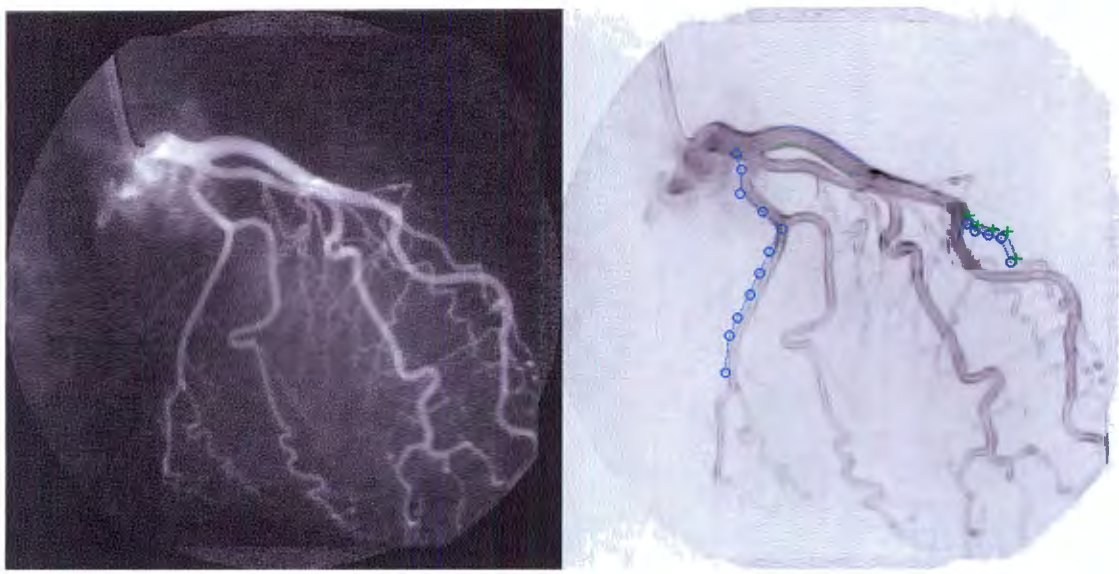


Figura 46. Contornos locales definidos por medio de B-splines

En la imagen anterior resulta un poco difícil notar la curva resultante. Los puntos de control están marcados por unas cruces azules para la curva de la izquierda y con círculos verdes para el caso de la curva de la derecha. En la imagen anterior no se aprecia la precisión, por lo que a continuación mostramos un acercamiento:

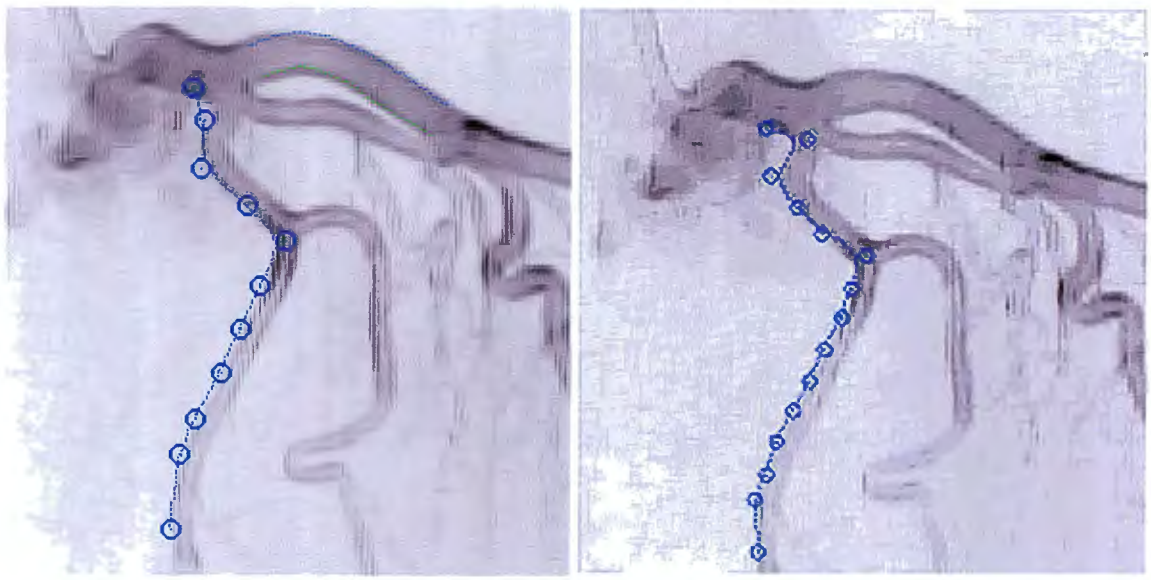


Figura 47. Once puntos de control (izquierda) vs quince puntos de control (derecha)

En la imagen de la izquierda (figura 47) podemos notar cómo en la parte superior vemos que no esta bien definido el contorno por la snake interpolada por los 11 puntos de control que se inicializaron. En la imagen de la derecha podemos ver cómo el contorno está perfectamente bien definido usando 15 puntos de control. El problema es que mientras más puntos agreguemos a la selección de puntos iniciales, el tiempo de cómputo aumenta de manera muy significativa.

En la imagen que se muestra a continuación podemos ver, de manera cualitativa que el método funciona de manera muy satisfactoria. Recordemos que cada curva tiene su propio conjunto de puntos de control de inicialización.

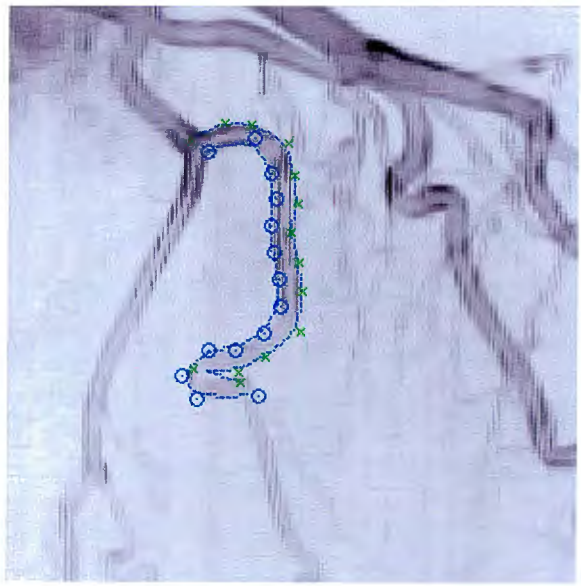


Figura 48. Estructura local bien definida

Una vez que la estructura local se encuentra bien definida, es posible pensar en el uso de otros algoritmos para la detección de patologías como, por ejemplo, la estenosis.

4.4 Estimación de Estenosis Aparente

Se simularon seis formas rectilíneas con estrechamientos de diferente grado. Se detectaron los contornos mediante el algoritmo de B-splines, luego se realizó la correspondencia entre los dos contornos y se calculó la Traza (en azul, en figura 49). A partir de la traza se estimaron los diámetros mínimos y máximos y se cuantificaron los diferentes grados de estenosis (**Tabla 1**).

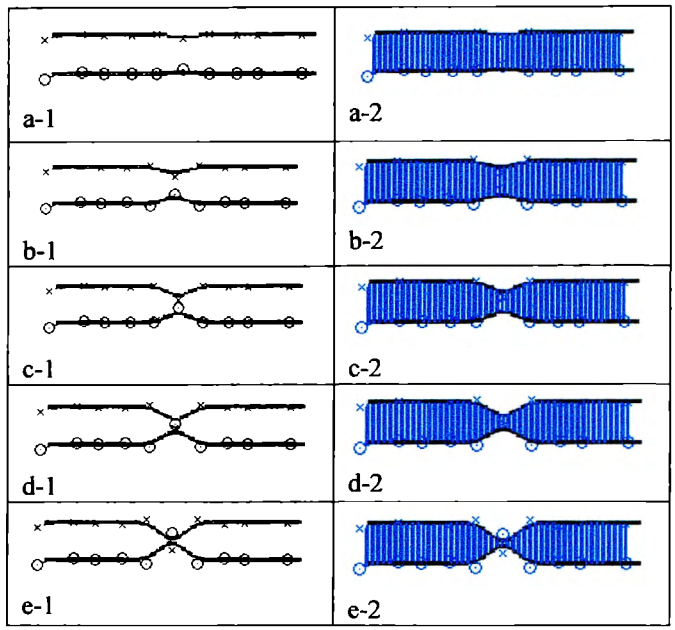


Figura 49. Simulaciones de arterias con diferentes grados de estenosis

	Ls(%)	*As(%)*	*Lr pixeles*	*Lm pixels*
a	10,4	19,8	18.8	21
b	29	50	14.8	21
c	48	73	10.8	21
d	67	89	6.8	21
e	86	98	2.8	21

Tabla 1. Porcentaje de estenosis para datos simulados

Se escogieron diferentes casos en imágenes reales y se caracterizaron diámetros mínimos y máximos. Los resultados de la caracterización no presentan estenosis severas (As superior a 50%) como se muestra en la **Tabla 2**.

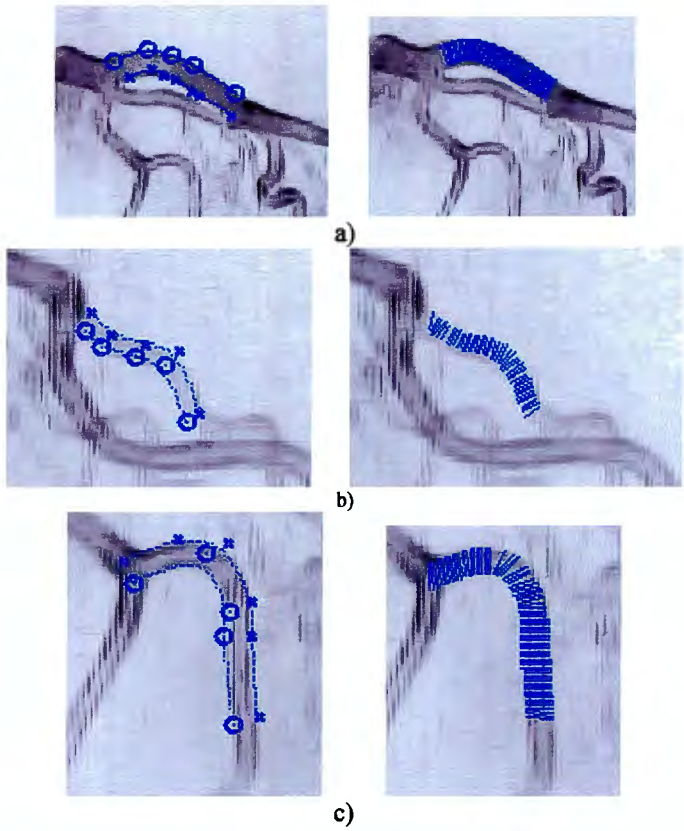


Figura 50. Determinación de porcentajes de estenosis para imágenes angiográficas reales

	Ls (%)	As(%)	Lr pixels	Lm pixels
a	12%	22%	14.42	16.41
b	17%	32%	5.7	6.9
c	11%	21%	9.41	10.6

Tabla 2. Porcentaje de estenosis para imágenes reales

5 Conclusiones y Perspectivas

En la literatura se proponen diferentes métodos para la segmentación de las imágenes, sin embargo, nosotros nos enfocamos en tres de los más característicos.

El gran valor que podemos asociar a este trabajo es que se presenta una comparación cualitativa de los tres algoritmos implementados. No se encuentran en la literatura estudios de comparación matemática de las técnicas de segmentación, solamente se pueden encontrar estudios sobre cada algoritmo, pero ningún documento que los compare entre sí y que pueda ayudar a escoger cualquiera de éstos para avanzar en estudios angiográficos que, para su consecución, requieran una segmentación previa.

Todos los algoritmos funcionaron correctamente y pudieron ser comparados de manera cualitativa. Sin embargo el método que se escogió fue el de B-splines en combinación con un banco de filtros S-Gabor debido a que es un método local a diferencia de momentos geométricos y wavelets que son globales. Se escogió el método local debido a que para la medición de las estenosis es mucho mejor analizar una sección de la imagen y no toda en general.

En cuanto a la caracterización de patologías se lograron mediciones cuantitativas. Los resultados sobre las imágenes reales son tan buenos como en las arterias simuladas. Se detectó el porcentaje de estenosis real de una sección en la imagen real.

Los resultados de los métodos de segmentación fueron validados de manera cualitativa, pero hemos dejado sentadas las bases para poder desarrollar una comparación cuantitativa.

El contexto general de un sistema de ayuda al cardiólogo basado en angiografía cardiaca tiene como etapas: la segmentación, reconstrucción en 3D de un sistema vascular cardiaco y análisis tanto de movimiento como de patologías. Este trabajo se enfocó principalmente en la comparación de los métodos de segmentación dado que la reconstrucción de un sistema vascular en 3D depende fundamentalmente de esta etapa.

Una vez que se tienen ambas validaciones y queda completamente demostrada su efectividad y robustez, se puede pensar en la reconstrucción de 3D y por consiguiente

se abre la posibilidad de llevar a cabo otras aplicaciones como son un análisis de movimiento o incluso la caracterización de otras patologías.

6 Fuentes

[1] TOUMOULIN C., COLLOREC R., COATRIEUX J.L., "*Vascular networks segmentation in substraction angiograms : a comparative study*", Med. Inform., 1990,15(4), pp 333-341

[2] RUAN S., BRUNO A., COATRIEUX J.L., " *Estimation de movement 3D en coronarographie*", Actes 13eme Colloque GRETSI, Juan les Pins, sept 1991, pp 801-904.

[3] LECORNU L., TOUMOULIN C., ROUX C., SENHADJI L., "*Using SVD and analyzing wavelets for the analysis and the segmentation of digital subtraction angiography images*", Proceedings Computer in cardiology, Oct. 1992, pp 655-658

[4] KLEIN A, FORESTER L., AMINI A., "*Quantitative coronary angiography with deformable spline models*", IEEE Transaction on Medical Imaging, 1997,16(5), pp 468-482

[5] BEMMEL C., SPREEUWERS L., VIERGEVER M., NIESSEN W., "*Level set based artery vein separation in blood pool agent CE-MR angiograms*", IEEE Trans. on Medical Imaging, 2003, 22(10), pp1224-1234

[6] SATO Y., ARAKI T., HANAYAMA M., NAITO H., TAMURA S., " *A view point determination system for stenosis diagnosis and quantification in coronary angiographic image acquisition*", IEEE Trans. on Medical Imaging , 1998,17(1), pp 121-137.

[7] KASS M., WITKIN A., TERZOPOULOS D., "*Snake: Active contours models*". Visual Computer.1988, pp 321-331.

[8] TERZOPOULOUS D., FLEISHER K., "Deformable models", Visual Computer, 4(6), pp 306-331, 1998

[9] MCINERNEY T., TERZOPOULOS D.: "*Deformable models in medical image analysis: a survey*"; Medical Image Analysis, vol. 1(2), pp.91-108, 1996.

[10] NAVARRO G., "*A guided tour ton approximate string matching*", ACM Computing survey,2001, 33 (1),pp 31-88

[12] WAGNER R.A., FISCHER M.J., "*The string to string correction problem* ", *Journal of ACM*, 21, 1974 :168-173

[13] TOUMOULIN C., BRIEVA J., BELLANGER J-J., SHU H.: "*String matching techniques for high level primitive formation in 2-D vascular imaging*", IEEE Transaction on Image and Technology in Medicine, 7(4), 2003, pp 291-301.

[14] LYVERS, et. Al. "*Subpixel Measurements Using a Moment-Based Edge Operator*", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 11, No 12, diciembre 1989.

[15] XU D.H., SHU H.Z.: BAO X.D., ZHANG H., DILLENSEGER J.L., LUO L.M.: "*A parabola fitting edge operator by the moments*". Engineering in Medicine and Biology Society, 2003. Proceedings of the 25th Annual International Conference of the IEEE , Volume: 1 , Sept. 17-21, 2003 Pages:653 – 656

[16] BRIEVA J. "*Mise en correspondence de lignes en angiographie cardiaqu*" Tesis. Université de Rennes 1. Febrero 2001.

[17] http://www.avera.org/adam/esp_imagepages/9156.htm

7 Anexos

Anexo A

Código para generar la arteria simulada con forma recta

```
  %TEST RECTA CON NIVELES DE GRIS CIRCULAR
close all;
clear all;

col1=120;
col2=400;
row1=100;
row2=120;
 region q no me interesa
    % %RELLENADO DE MATRIZ 512 X 512 CON NIVELES DE GRIS COLOR NEGRO
for I=1:512
    for J=1:512
        TESTcurva(I,J)=0;
    end
end
   %%%%%%.  %%%%%%%%%%%%          NIVEL DEL GRIS CIRCULAR

k=1;
for y = (row2+row1)/2-row1:-1:0
z(k) = sqrt(((row2+row1)/2-row1)^2-y^2);
k=k+1;
end

for y= 1:(row2+row1)/2-row1
z(k) = sqrt(((row2+row1)/2-row1)^2-y^2);
k=k+1;
end


for J=col1:col2
k=1;
```

```
for I=row1:row2
        TESTrecta(I,J)=20*z(k);%MAX NIVEL DE GRIS = 200
k=k+1;
end
end


figure,
imagesc(TESTrecta);
colorbar;
axis off;
axis ij;
axis manual;
title(['RECTA DE PRUEBA']);
axis([0 512 0 512]);
axis square;
colormap(gray);


fid = fopen('testrecta10','w');
for I=1:512
for J=1:512
fwrite(fid,TESTrecta(I,J),'char',0);
end
end
fclose(fid);



% ATENCION!!!!! CON ESTO SE CREA EL CONTORNO DE LA RECTA DE PRUEBA
for c=col1:col2
r=row1;
L1= line (c,r);
set(L1,'color','yellow');
set (L1,'lineWidth',1);
end

for c=col1:col2
r=row2;
L1= line (c,r);
set(L1,'color','yellow');
set (L1,'lineWidth',1);
end

for r=row1:row2
c=col1;
L1= line (c,r);
set(L1,'color','yellow');
set (L1,'lineWidth',1);
end

for r=row1:row2
c=col2;
L1= line (c,r);
set(L1,'color','yellow');
set (L1,'lineWidth',1);
end
```

Anexo B

Código para generar la arteria simulada con forma curva

```
%=================GENERACION DE LA CURVA PARA LAS PRUEBAS
close all;
clear all;
x=0:0.1:50;
y=50*exp(-((x.^2)/200));
figure,
plot (x,y);

r=10;

% GENERACION DE ARREGLOS CON LOS PUNTOS DE LA CURVA ORIGINAL i= 501
i=1;
for x=0:0.1:50
xi(i)=x;
yi(i)=50*exp(-(x^2/200));
i=i+1;
end

% GENERACION DE ARREGLOS CON LOS PUNTOS ANTERIOR Y SIGUIENTE PARA LA
TANGENTE
for i=2:500;
xti(i)=xi(i-1)-xi(i+1);
yti(i)=yi(i-1)-yi(i+1);
end

%CALCULO DE LA PENDIENTE DE LA CURVA ORIGINAL Y DE LA RECTA
PERPENDICULAR
%for i=2:500;
for i=20:500;
m(i) = (yi(i+1)-yi(i-1))/(xi(i+1)-xi(i-1));
mp(i) = -1/m(i);
teta(i) = atan (mp(i));

% CALCULO DE LAS COORDENADAS DE LAS CURVAS CONTORNOS
yr1(i) = r*sin(teta(i))+yi(i);
%xr1(i) = r*cos(teta(i))+xi(i);
yr2(i) = yi(i)-r*sin(teta(i));
xr2(i) = xi(i)-r*cos(teta(i));
%end


%CALCULO DE LAS COORDENADAS DE LAS CURVAS CONTORNOS CON VALORES
DISCRETOS
yr1(i) = round(r*sin(teta(i))+yi(i));
xr1(i) = round(r*cos(teta(i))+xi(i));
yr2(i) = round(yi(i)-r*sin(teta(i)));
xr2(i) = round(xi(i)-r*cos(teta(i)));
end


%DIBUJO DE LINEAS DESDE UN CONTORNO HASTA EL OTRO
for i=20:1:500
```

```
A = [xr1(i) xr2(i)];
B = [yr1(i) yr2(i)];
line (A,B,'LineWidth',1,'Color',[0 1 0]);
end


%DIBUJO DE LAS CURVAS CONTORNOS
for i=20:500
L1= line (xr1(i),yr1(i));
L2= line (xr2(i),yr2(i));
set(L1,'color','blue');
set (L1,'lineWidth',1);
set(L2,'color','blue');
set (L2,'lineWidth',1);
end


%%%%CON ESTO SE GENERAN LOS PUNTOS DE CADA RECTA PERPENDICULAR A LA
TANGENTE

%for i=20:2:500
for i=20:1:500
P1(1)=round(xr2(i));
P1(2)=round(yr2(i));

P2(1)=round(xr1(i));
P2(2)=round(yr1(i));

[points, numpoints] = interpoladorlineal(P1, P2);

for k=1:numpoints
L3= line (points(k),points(k+numpoints));
set(L3,'color','black');
set (L3,'lineWidth',1);
end
end

%%%%%%RELLENADO DE MATRIZ 512 X 512 CON NIVELES DE GRIS COLOR NEGRO
for I=1:512
    for J=1:512
        TESTcurva(I,J)=0;
    end
end

%%% NIVELES DE GRIS CIRCULAR

q=1;
for i=20:1:230 %%% ESTA PARTE ES PARA LOS PUNTOS HASTA LA CURVA DE
ABAJO======CICLO FOR 1000

P1(1)=round(xr2(i));
P1(2)=round(yr2(i));

P2(1)=round(xr1(i));
P2(2)=round(yr1(i));

[points, numpoints] = interpoladorlineal(P1, P2);
```

```
numerodepuntos(i)=numpoints;

g=1;
%%%  CON ESTO SACO LAS COORDENADAS DE LOS PUNTOS Q FORMAN LAS RECTAS
PERPENDICULARES A %%%  LAS TANGENTES
for k=1:numpoints
ex(g)=points(k);
wye(g)=points(k-numpoints);
g=g+1;
end


%%%CON ESTO SE ASIGNAN VALORES DE GRIS DE FORMA CIRCULAR

if (numpoints==14)||(numpoints==16)||(numpoints==18)||(numpoints==20)
centro = (numpoints/2);
centro1 = (numpoints/2)+1;
for g=1:centro
%WYE(q) = real (sqrt((10)^2-(ex(numpoints)-ex(g))^2));
WYE(q) = (sqrt((10)^2-(ex(g)-ex(centro))^2));
%TESTcurva(ex(g)+300,wye(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
TESTcurva(wye(g)+300,ex(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10


q=q+1;
end

for g=centro1:1:numpoints
WYE(q) = (sqrt((10)^2-(ex(g)-ex(centro1))^2));%%%%  MAX NIVEL DE  GRIS
20*10
%WYE(q) = (sqrt((10)^2-(ex(centro)-ex(g))^2));
TESTcurva(wye(g)+300,ex(g)+300)=20*WYE(q);
%TESTcurva(ex(g)+300,wye(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
q=q+1;
end


else
centro = ceil(numpoints/2);
for g=1:centro
%WYE(q) = real (sqrt((10)^2-(ex(numpoints)-ex(g))^2));
WYE(q) = (sqrt((10)^2-(ex(g)-ex(centro))^2));

TESTcurva(wye(g)+300,ex(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
%TESTcurva(ex(g)+300,wye(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
q=q+1;
end

for g=centro+1:1:numpoints
WYE(q) = (sqrt((10)^2-(ex(g)-ex(centro))^2));
%WYE(q) = (sqrt((10)^2-(ex(centro)-ex(g))^2));
```

```
TESTcurva(wye(g)+300,ex(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
%TESTcurva(ex(g)+300,wye(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
q=q+1;
end

end %%% END DEL IF
end   %%%  END CICLO FOR 1000


%==================================================================
============
for i=231:1:500 %%% ESTA PARTE ES PARA LOS PUNTOS HASTA LA CURVA DE
ABAJO======CICLO FOR 1000
%for i=150:1:150 %%% CICLO FOR 1000

P1(1)=round(xr2(i));
P1(2)=round(yr2(i));

P2(1)=round(xr1(i));
P2(2)=round(yr1(i));

[points, numpoints] = interpoladorlineal(P1, P2);
numerodepuntos(i)=numpoints;

g=1;
%    CON ESTO SACO LAS COORDENADAS DE LOS PUNTOS Q FORMAN LAS RECTAS
PERPENDICULARES A %%% LAS TANGENTES
for k=1:numpoints
ex(g)=points(k);
wye(g)=points(k+numpoints);
g=g+1;
end


%%% CON ESTO SE ASIGNAN VALORES DE GRIS DE FORMA CIRCULAR

if (numpoints==14)||(numpoints==16)||(numpoints==18)||(numpoints==20)
centro = (numpoints/2);
centro1 = (numpoints/2)+1;
for g=1:centro
%WYE(q) = real (sqrt((10)^2-(ex(numpoints)-ex(g))^2));
WYE(q) = (sqrt((10)^2-(wye(g)-wye(centro))^2));

TESTcurva(wye(g)+300,ex(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
TESTcurva(ex(g)+300,wye(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
q=q+1;
end

for g=centro1:1:numpoints
WYE(q) = (sqrt((10)^2-(wye(g)-wye(centro1))^2));%%%%  MAX NIVEL DE
GRIS 20*10
%WYE(q) = (sqrt((10)^2-(ex(centro)-ex(g))^2));
TESTcurva(wye(g)+300,ex(g)+300)=20*WYE(q);
```

```
%TESTcurva(ex(g)+300,wye(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
q=q+1;
end


else
centro = ceil(numpoints/2);
for g=1:centro
 WYE(q) = real (sqrt((10)^2-(ex(numpoints)-ex(g))^2));
WYE(q) = (sqrt((10)^2-(wye(g)-wye(centro))^2));

TESTcurva(wye(g)+300,ex(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
%TESTcurva(ex(g)+300,wye(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
q=q+1;
end

for g=centro+1:1:numpoints
WYE(q) = (sqrt((10)^2-(wye(g)-wye(centro))^2));
%WYE(q) = (sqrt((10)^2-(ex(centro)-ex(g))^2));
TESTcurva(wye(g)+300,ex(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
TESTcurva(ex(g)+300,wye(g)+300)=20*WYE(q);%%%%  MAX NIVEL DE  GRIS
20*10
q=q+1;
end

end %%% END DEL IF


end   %%%  END CICLO FOR 1000

%%%=========PARA CUBRIR LOS PUNTOS DE LA IMAGEN Q NO SE HAN LLENADO,
HAGO UN BARRIDO %%%%SOBRE TODOS LOS PIXELES
%%%====DE LA IMAGEN Y CHECO POR PUNTOS NEGROS Q TENGAN 4 VECINOS
(ARRIBA, ABAJO, IZQ Y %%%%DER) Q NO
 %%%====SEAN NEGROS Y A ESE LE PONGO UN NIVEL DE GRIS IGUAL AL PROMEDIO
DE LOS VECINOS

for I=250:400
    for J=250:400
        if TESTcurva(I-1,J)~=0 && TESTcurva(I+1,J)~=0 && TESTcurva(I,J-
1)~=0 && TESTcurva(I,J+1)~=0
        nivel = (TESTcurva(I-1,J) + TESTcurva(I+1,J) + TESTcurva(I,J-1)
+ TESTcurva(I,J+1))/4;
        TESTcurva(I,J)=round(nivel);
end
    end
end

figure,
imagesc(TESTcurva);
colorbar;
axis off;
axis ij;
```

```
axis manual;
title(['CURVA DE PRUEBA']);
axis([0 512 0 512]);
axis square;
colormap(gray);

%%========== DIBUJO DE LOS PUNTOS DE CONTORNO SOBRE LA IMAGEN FINAL
%DIBUJO DE LAS CURVAS CONTORNOS
for i=20:500
L1= line (xr1(i)+300,yr1(i)+300);
L2= line (xr2(i)+300,yr2(i)+300);
set(L1,'color','red');
set (L1,'lineWidth',1);
set(L2,'color','red');
set (L2,'lineWidth',1);
end

    guardar imagen en formato binario
fid = fopen('testcurva','w');
for I=1:512
for J=1:512
fwrite(fid,TESTcurva(I,J),'char',0);
end
end
fclose(fid);
```

Anexo C

Código de implementación del método de wavelets

```
 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!METODO    WAVELETS!!!!!!!!!!!!!!!!!!!

%%%%PARA Q FUNCIONE EL METODO DE WAVELET, SE DEBE DE LLAMAR LA FUNCION
 %%%"WAVELETS" Y MANDARLE DE PARAMETROS EL NOMBRE DE LA IMAGEN Q SE
DESEA
%%%%PROCESAR, Y LOS 5 VALORES DE N...LOS VALORES POR DEFAULT DE LAS Ns
SON 6.8.11.15.20
%%%%EL METODO AL FINAL ABRE
%%%%UNA VENTANA PARA DESPLEGAR LA IMAGEN PERO SE LE PUEDE COMENTAR ESO
PARA
%%%%Q SOLO REGRESE EL VALOR EN BINARIO DE LA IMAGEN....TAMBIEN TIENE AL
%%%%FINAL LA RUTINA PARA Q GUARDE EN ARCHIVO LA MATRIZ DE VALORES
BINARIOS
 %%%%DE LA IMAGEN PERO ESO ESTA COMENTADO, DESPUES HAY Q DESCOMENTARLO


%%%%LA SINTAXIS CON LA Q SE DEBE ABRIR EL METODO WAVELET ES:

  %%                  fid = fopen('seg2','r');    'seg2' es el archivo q
quiero procesar
%                     for I = 1:512
%                       Imaor(I,1:1:512)=fread(fid,[1,512],'char',0);
                      end
%%%                   fclose(fid);
    %                 imagenWAVELET = wavelets(Imaor,N1,N2,N3,N4,N5);

function [ImagenFinal]=wavelets(Imaor,N1,N2,N3,N4,N5)
 VERTICAL
VER6=waveletVER(Imaor,N1);
VER8=waveletVER(Imaor,N2);
VER11=waveletVER(Imaor,N3);
VER15=waveletVER(Imaor,N4);
VER20=waveletVER(Imaor,N5);

%HORIZONTAL
HOR6=waveletHOR(Imaor,N1);
HOR8=waveletHOR(Imaor,N2);
HOR11=waveletHOR(Imaor,N3);
HOR15=waveletHOR(Imaor,N4);
HOR20=waveletHOR(Imaor,N5);

%DIAGONAL1
DIAG16=waveletDIAG1(Imaor,N1);
DIAG18=waveletDIAG1(Imaor,N2);
DIAG111=waveletDIAG1(Imaor,N3);
DIAG115=waveletDIAG1(Imaor,N4);
DIAG120=waveletDIAG1(Imaor,N5);

  DIAGONAL2
DIAG26=waveletDIAG2(Imaor,N1);
```

```
DIAG28=waveletDIAG2(Imaor,N2);
DIAG211=waveletDIAG2(Imaor,N3);
DIAG215=waveletDIAG2(Imaor,N4);
DIAG220=waveletDIAG2(Imaor,N5);

% MAXIMA
% N=6
for row = 32:1:481
    for col = 32:1:481
      A6 = VER6 (row , col);
      B6 = HOR6 (row , col);
      C6 = DIAG16 (row , col);
      D6 = DIAG26 (row , col);

      Z = [A6 B6 C6 D6];
      M = max(Z);
      ImaMax6 (row , col) = M;
    end
end

  N=8
for row = 32:1:481
    for col = 32:1:481
      A8 = VER8 (row , col);
      B8 = HOR8 (row , col);
      C8 = DIAG18 (row , col);
      D8 = DIAG28 (row , col);

      Z = [A8 B8 C8 D8];
      M = max(Z);
      ImaMax8 (row , col) = M;
    end
end

% N=11
for row = 32:1:481
    for col = 32:1:481
      A11 = VER11 (row , col);
      B11 = HOR11 (row , col);
      C11 = DIAG111 (row , col);
      D11 = DIAG211 (row , col);

      Z = [A11 B11 C11 D11];
      M = max(Z);
      ImaMax11 (row , col) = M;
    end
end

% N=15
for row = 32:1:481
    for col = 32:1:481
      A15 = VER15 (row , col);
      B15 = HOR15 (row , col);
      C15 = DIAG115 (row , col);
      D15 = DIAG215 (row , col);

      Z = [A15 B15 C15 D15];
```

```
      M = max(Z);
      ImaMax15 (row , col) = M;
    end
end

  N=20
for row = 32:1:481
    for col = 32:1:481
      A20 = VER20 (row , col);
      B20 = HOR20 (row , col);
      C20 = DIAG120 (row , col);
      D20 = DIAG220 (row , col);

      Z = [A20 B20 C20 D20];
      M = max(Z);
      ImaMax20 (row , col) = M;
    end
end

%BINARIZACION
col = 1;
row = 1;
 for row = 32:1:481
    for col = 32:1:481
        if ImaMax6 (row , col) > 28
            ImaMax6BIN (row , col) = 0;
        else ImaMax6BIN (row , col) = 255;
        end

    end
end

col = 1;
row = 1;
 for row = 32:1:481
    for col = 32:1:481
        if ImaMax8 (row , col) > 40
            ImaMax8BIN (row , col) = 0;
        else ImaMax8BIN (row , col) = 255;
        end

    end
end

col = 1;
row = 1;
 for row = 32:1:481
    for col = 32:1:481
        if ImaMax11 (row , col) > 55
            ImaMax11BIN (row , col) = 0;
        else ImaMax11BIN (row , col) = 255;
        end

    end
end

col = 1;
```

```
row = 1;
 for row = 32:1:481
    for col = 32:1:481
        if ImaMax15 (row , col) > 80
            ImaMax15BIN (row , col) = 0;
        else ImaMax15BIN (row , col) = 255;
        end
      end
end

col = 1;
row = 1;
 for row = 32:1:481
    for col = 32:1:481
        if ImaMax20 (row , col) > 93
            ImaMax20BIN (row , col) = 0;
        else ImaMax20BIN (row , col) = 255;
        end
    end
end

  OR FINAL
col = 1;
row = 1;
 for row = 32:1:481
    for col = 32:1:481
      BIN6 = ImaMax6BIN (row , col);
      BIN8 = ImaMax8BIN (row , col);
      BIN11 = ImaMax11BIN (row , col);
      BIN15 = ImaMax15BIN (row , col);
      BIN20 = ImaMax20BIN (row , col);

      Zfinal = [BIN6 BIN8 BIN11 BIN15 BIN20];
      ImagenFinal (row , col) = min(Zfinal);
    end
end

col = 1;
row = 1;
 for row = 1:31
     for col = 1:512
         ImaMax6BIN (row, col)= 255;
         ImaMax8BIN (row, col)= 255;
         ImaMax11BIN (row, col)= 255;
         ImaMax15BIN (row, col)= 255;
         ImaMax20BIN (row, col)= 255;
         ImagenFinal (row, col)= 255;
     end
 end

  for row = 482:512
     for col = 1:512
         ImaMax6BIN (row, col)= 255;
         ImaMax8BIN (row, col)= 255;
         ImaMax11BIN (row, col)= 255;
         ImaMax15BIN (row, col)= 255;
         ImaMax20BIN (row, col)= 255;
```

```
            ImagenFinal (row, col)= 255;
        end
    end

    for row = 1:512
        for col = 1:31
            ImaMax6BIN (row, col)= 255;
            ImaMax8BIN (row, col)= 255;
            ImaMax11BIN (row, col)= 255;
            ImaMax15BIN (row, col)= 255;
            ImaMax20BIN (row, col)= 255;
            ImagenFinal (row, col)= 255;
        end
    end

    for row = 1:512
        for col = 482:512
            ImaMax6BIN (row, col)= 255;
            ImaMax8BIN (row, col)= 255;
            ImaMax11BIN (row, col)= 255;
            ImaMax15BIN (row, col)= 255;
            ImaMax20BIN (row, col)= 255;
            ImagenFinal (row, col)= 255;
        end
    end

figure,
imagesc(ImagenFinal);
colorbar;
axis off;
axis ij;
axis manual;
title(['IMAGEN FINAL....WAVELETS']);
axis([0 512 0 512]);
axis square;
colormap(gray);

  %CON ESTO SE GUARDA LA IMAGEN FINAL SEGMENTADA EN FORMATO BINARIO
  fid = fopen('waveletFINALTESTcurvabin','w');
  for I=1:512
                for J=1:512
                  fwrite(fid,ImagenFinal(I,J),'char',0);
                end
  end
  fclose(fid);
 =========FILTRADO VERTICAL ==========================================
function ImaSegm=waveletVER(ImaOr, N)
F=1/(2*N);
kx=2;
j=1;
for x = -N:1:N
    Phi(j)=(1+cos(2*pi*F*x))*exp(-i*2*pi*kx*F*x);
    j=j+1;
end
for I=32:481
    for J=32:481
        sumaconvolucion=0;
```

```
        for K=-N:N
            sumaconvolucion=sumaconvolucion+Phi(K+N+1)*ImaOr(I-K,J);
        end
        ImaSegm(I,J)=real(sumaconvolucion);
    end
end
%=========FILTRADO HORIZONTAL
==========================================
function ImaSegm=waveletHOR(ImaOr, N)
F=1/(2*N);
kx=2;
j=1;
for x = -N:1:N
    Phi(j)=(1+cos(2*pi*F*x))*exp(-i*2*pi*kx*F*x);
    j=j+1;
end
for I=32:481
    for J=32:481
        sumaconvolucion=0;
        for K=-N:N
            sumaconvolucion=sumaconvolucion+Phi(K+N+1)*ImaOr(I,J-K);
        end
        ImaSegm(I,J)=real(sumaconvolucion);
    end
end
%==========FILTRADO DIAGONAL 1
=========================================
function ImaSegm=waveletDIAG1(ImaOr, N)
F=1/(2*N);
kx=2;
j=1;
for x = -N:1:N
    Phi(j)=(1+cos(2*pi*F*x))*exp(-i*2*pi*kx*F*x);
    j=j+1;
end
for I=32:481
    for J=32:481
        sumaconvolucion=0;
        for K=-N:N
            sumaconvolucion=sumaconvolucion+Phi(K+N+1)*ImaOr(I-K,J+K);
        end
        ImaSegm(I,J)=real(sumaconvolucion);
    end
end
%=========FILTRADO DIAGONAL 2
==========================================
function ImaSegm=waveletDIAG2(ImaOr, N)
F=1/(2*N);
kx=2;
j=1;
for x = -N:1:N
    Phi(j)=(1-cos(2*pi*F*x))*exp(-i*2*pi*kx*F*x);
    j=j+1;
end
for I=32:481
    for J=32:481
        sumaconvolucion=0;
```

```
        for K=-N:N
            sumaconvolucion=sumaconvolucion+Phi(K+N+1)*ImaOr(I-K,J-K);
        end
        ImaSegm(I,J)=real(sumaconvolucion);
    end
end
```

Anexo D

Código de implementación del método de filtros de Gabor

```
%!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!METODO SGABOR 2D!!!!!!!!!!!!!!!!!!!!

%%%%PARA Q FUNCIONE EL METODO DE SGABOR, SE DEBE DE LLAMAR LA FUNCION
% %%"SGABOR2D" Y MANDARLE DE PARAMETROS EL NOMBRE DE LA IMAGEN Q SE
DESEA
%%%%PROCESAR, EL METODO AL FINAL ABRE UNA VENTANA PARA DESPLEGAR
%%%% LA IMAGEN PERO SE LE PUEDE COMENTAR ESO PARA
%%%%Q SOLO REGRESE EL VALOR EN BINARIO DE LA IMAGEN....TAMBIEN TIENE AL
%%%%FINAL LA RUTINA PARA Q GUARDE EN ARCHIVO LA MATRIZ DE VALORES
BINARIOS
%%%%DE LA IMAGEN PERO ESO ESTA COMENTADO, DESPUES HAY Q DESCOMENTARLO
%%%%LA SINTAXIS CON LA Q SE DEBE ABRIR EL METODO SGABOR2D ES:

    %                fid = fopen('seg2','r');    %%%'seg2' es el archivo
q quiero procesar
    %                for I = 1:512
                       Imaor(I,1:1:512)=fread(fid,[1,512],'char',0);
                     end
    %                fclose(fid);
    %                imagenSGABOR2D = SGABOR2D(Imaor);

function [GABORFINALROTADA]=SGABOR2D(Imaor)
N=10;
lamda = -0.4;
global Imaor
global lamda
global N


teta= 0*2*pi/360;
maskara1=mask12Dsgaborrotada(teta);
maskara2=mask22Dsgaborrotada(teta);
for I=N+1:512-(N+1)
    for J=N+1:512-(N+1)
        teta0(I,J)=sqrt ((maskara1(I,J))^2 + (maskara2(I,J))^2);
    end
end


teta= 45*2*pi/360;
maskara1=mask12Dsgaborrotada(teta);
maskara2=mask22Dsgaborrotada(teta);
for I=N+1:512-(N+1)
    for J=N+1:512-(N+1)
        teta45(I,J)=sqrt ((maskara1(I,J))^2 + (maskara2(I,J))^2);
    end
end


teta= 90*2*pi/360;
maskara1=mask12Dsgaborrotada(teta);
```

```
maskara2=mask22Dsgaborrotada(teta);
for I=N+1:512-(N+1)
    for J=N+1:512-(N+1)
        teta90(I,J)=sqrt ((maskara1(I,J))^2 + (maskara2(I,J))^2);
    end
end
teta= 135*2*pi/360;
maskara1=mask12Dsgaborrotada(teta);
maskara2=mask22Dsgaborrotada(teta);
for I=N+1:512-(N+1)
    for J=N+1:512-(N+1)
        teta135(I,J)=sqrt ((maskara1(I,J))^2 + (maskara2(I,J))^2);
    end
end


% MAXIMIZAZION
for I=N+1:512-(N+1)
    for J=N+1:512-(N+1)
      Z = [teta0(I,J) teta45(I,J) teta90(I,J) teta135(I,J)];
      M = max(Z);
      GABORFINALROTADA (I,J) = M;
    end
end

  DESPLEGAR IMAGENES
%IMAGEN ORIGINAL
figure,
imagesc(Imaor);
colorbar;
axis ij;
axis manual;
axis([0 512 0 512]);
title(['IMAGEN ORIGINAL']);
axis square;
colormap(gray);
axis off;


%llenado de la imagen final para que sea de 512 x 512
for I=1:N
    for J=1:512
        GABORFINALROTADA(I,J)=0;
    end
end
for I=512-N:512
    for J=1:512
        GABORFINALROTADA(I,J)=0;
    end
end
for I=1:512
    for J=1:N
        GABORFINALROTADA(I,J)=0;
    end
end
for I=1:512
    for J=512-N:512
```

```
        GABORFINALROTADA(I,J)=0;
    end
end

figure,
imagesc(GABORFINALROTADA);
colorbar;
axis off;
axis ij;
axis manual;
title(['Sgabor 2D con rotaciones']);
axis([0 512 0 512]);
axis square;
colormap(gray);


%CON ESTO SE GUARDA LA IMAGEN FINAL SEGMENTADA
fid = fopen('SGABOR2D','w');
for I=1:512
                for J=1:512
                  fwrite(fid,GABORFINALROTADA(I,J),'char',0);
                end
end
fclose(fid);


%==============MASCARA 1
S(X)=========================================================
function ImaSegm=mask12Dsgaborrotada(teta)
global Imaor
global lamda
global N
sigmax = 1.5;
k = 0.9;
v0 = 0.39;
sigmay = 2;
for x = -N:N
    for y = -N:N
        xr=x*cos(teta)-y*sin(teta);
                yr=x*sin(teta)+y*cos(teta);
        S(x+N+1,y+N+1) = exp(-0.5*(xr^2 / sigmax^2 + yr^2 /
sigmay^2))*sin(2*pi*v0*xr*(k*exp(-lamda*(xr/sigmax)^2)));
    end
end

for I=N+1:512-(N+1)
    for J=N+1:512-(N+1)
        sumaconvolucion1=0;
        for x=-N:N
            for y=-N:N

sumaconvolucion1=sumaconvolucion1+S(x+N+1,y+N+1)*Imaor(I-x,J-y);
            end
        end
        mask1(I,J)=(sumaconvolucion1);
    end
end
```

```
for I=N+1:512-(N+1)
    for J=N+1:512-(N+1)
        ImaSegm(I,J)=mask1(I,J);
    end
end

%==============MASCARA 2 C(X)
=====================================================
function ImaSegm=mask22Dsgaborrotada(teta)
global Imaor
global lamda
global N
sigmax = 0.9;
k = 0.9;
v0 = 0.39;
sigmay = 2;
for x = -N:N
    for y = -N:N
        xr=x*cos(teta)-y*sin(teta);
                yr=x*sin(teta)+y*cos(teta);
        C(x+N+1,y+N+1) = exp(-0.5*(xr^2 / sigmax^2 + yr^2 /
sigmay^2))*cos(2*pi*v0*xr*(k*exp(-lamda*(xr/sigmax)^2)));
    end
end
for I=N+1:512-(N+1)
    for J=N+1:512-(N+1)
        sumaconvolucion2=0;
        for x=-N:N
            for y=-N:N

sumaconvolucion2=sumaconvolucion2+C(x+N+1,y+N+1)*Imaor(I-x,J-y);
            end
        end
        mask2(I,J)=(sumaconvolucion2);
    end
end

for I=N+1:512-(N+1)
    for J=N+1:512-(N+1)
        ImaSegm(I,J)=mask2(I,J);
    end
end
```

Anexo E

Código de implementación del método de momentos geométricos

```
Main
clear all;
close all;

  AQUI SE INTRODUCE EL VALOR DE r

r = input('Que valor quiere de r?\n');

  AQUI SE DEFINE LA IMAGEN QUE SE VA A ABRIR.

fid = fopen('seg2','r');
for I = 1:512,
Imaor(I,1:1:512)=fread(fid,[1,512],'char',0);
end
fclose(fid);
Seg2Max=max(max(Imaor))
Seg2Min=min(min(Imaor))

 CON ESTO SE OBTIENEN LOS MOMENTOS DE LA IMAGEN
M00 = convIsraelb(Imaor,Ma00);
M01 = convIsraelb(Imaor,Ma01);
M10 = convIsraelb(Imaor,Ma10);
M20 = convIsraelb(Imaor,Ma20);
M02 = convIsraelb(Imaor,Ma02);
M11 = convIsraelb(Imaor,Ma11);

  Aqui se genera TETA

angulo = Teta(M01,M10);

 CON ESTO SE OBTIENTN LOS MOMENTOS PRIMADOS

M10p = M10ap(M10,M01);
M20p = M20ap(M10,M20,M01,M11,M02);
M02p = M02ap(M10,M20,M01,M11,M02);

% SE SACAN LOS PARAMETROS PARABOLICOS

M=parM(r,M00,M02p,M10p);
d=parD(r,M02p,M20p,M00,M10p);
mc=parmm(r,d,M);
b=parB(r,d,mc,M10p);

%MUESTRA LOS RESULTADOS FINALES

 Resultado = contorno(b,d,angulo);

%SE GRAFICAN LOS PARAMETROS DESEADOS
%GRAFICAMOS RESULTADO
```

```
figure,
imagesc(Resultado);
title('Resultado Final');
colorbar;
axis ij;
axis manual;
axis([0 512 0 512]);
axis square;
colormap(gray);
axis off;
```

FUNCIONES

Convolución

```
function [Ir]=convIsraelb(I,M);

Ir(1:1:512,1:1:512)=0;
for i=3:510;
for j=3:510;
    suma=0;
    for k=-2:2
    for l=-2:2
        suma=suma+M(k+3,l+3)*I(i-k,j-l);
    end
    end
Ir(i,j)=suma;
end
end
a=size(Ir)
```

```
M02P*****
%ESTE ARCHIVO REALIZA EL MOMENTO 02 PRIMO Y EN LA DIVISION POR CERO SE
DAN VALORES DE 255 CONSIDERANDO QUE UN NUMERO DIVIDIDO ENTRE CERO
%TIENDE A INFINITO Y SE REDONDEA AL ENTERO MAS ALTO DE LA IMAGEN EN
GRISES

%ABRO EL MOMENTO M10

function [M02p] = M02ap(M10,M20,M01,M11,M02)
for I=1:512
    for J=1:512
        if((M01(I,J)+M10(I,J))==0)
            M02p(I,J)=0;
        else
            M02p(I,J)=(1/(M01(I,J)^2+M10(I,J)^2))*(M01(I,J)^2*M20(I,J)-
2*M10(I,J)*M01(I,J)*M11(I,J)+M10(I,J)^2*M02(I,J));
        end
end
end


M10P
%CREACION DEL MOMENTO M10' QUE SIRVE PARA GENERAR EL OPERADOR L

% RAIZ(M01^2 + M10^2)
```

```
function [p] = M10ap(M10,M01)

% HAGO LAS OPERACIONES PARA OBTENER M10' EN LA VARIABLE p

for I=1:512
    for J=1:512
        p(I,J)=(M10(I,J)^2+M01(I,J)^2)^(.5);
    end
end

%CREACION DE M20'

function [Mx] = M20ap(M10,M20,M01,M11,M02)

%HAGO LA OPERACION PARA ENCONTRAR M20PRIMO EN LA VARIABLE Mx SE DEBE
NOTAR QUE SE DAN MUCHAS DIVISIONES ENTRE CERO, POR LO QUE SE LE PONE EL
IF SIN EMBARGO ESTO ALENTA MUCHO EL ALGORITMO

for I=1:512
    for J=1:512
        if((M01(I,J)+M10(I,J))==0)
            Mx(I,J)=0;
        else

Mx(I,J)=(M10(I,J)^2*M20(I,J)+2*M01(I,J)*M10(I,J)*M11(I,J)+M01(I,J)^2*M0
2(I,J))/(M01(I,J)^2+M10(I,J)^2);
    end
end
end

%ESTE PROGRAMA CALCULA EL PARAMETRO B DEL OPERADOR PARABOLICO A PARTIR
DE m, M10p y D.

function [b] = parB(r,d,mc,M10p)

for I=1:512
    for J=1:512
        if(mc(I,J)==r)
            b(I,J)=0;
        else
            b(I,J)=(15*M10p(I,J))/((2*(r^2-mc(I,J)^2)^0.5)*(mc(I,J)^2-
            2*mc(I,J)*d(I,J)-4*d(I,J)^2+5*r^2));
  end
  end
end

%Este archivo realiza el parametro d del momento parabolico con el
valor de r que se da en el main

function [d]=parD(r,M02p,M20p,M00,M10p);


for I=1:512
    for J=1:512
        if(M10p(I,J)==0)
            d(I,J)=0;
        else
```

```
        d(I,J)=((7*M20p(I,J)+5*M02p(I,J)-*r^2*M00(I,J))/(4*M10p(I,J)));
        end
    end
end

dMax = max(max(d))
dMin = min(min(d))
dPromedio = mean(mean(d))


%Este archivo realiza el parametro M del momento parabolico con el
valor de r que se introduce en la funcion

function [M] = parM(r,M00,M02p,M10p)

for I=1:512
    for J=1:512
        if(M10p(I,J)==0)
            M(I,J)=0;
        else
            M(I,J)=(r^2*M00(I,J)-4*M02p(I,J))/M10p(I,J);
        end
end
end

%parametor m minuscula de la eq de 3er orden

%Este archivo realiza el parametro d del momento parabolico con el
valor de r que propuse de r=1

function [mc] = parmm(r,d,M)

a=3;

for I=1:512
    for J=1:512
    b(I,J)=8*d(I,J)+M(I,J);
    c(I,J)=7*r^2+2*M(I,J)*d(I,J);
    x(I,J)=4*M(I,J)*d(I,J)^2-2*d(I,J)*r^2-5*M(I,J)*r^2;
    mc(I,J)=isreal(roots([a,b(I,J),c(I,J),x(I,J)]));%%aux
    end
end

% Este archivo genera el operador Teta obtenido por atan(M01/M10)

function [x] = Teta(M01,M10)

%HAGO LA OPERACION DE TETA
for k=1:512
    for l=1:512
        if(M10(k,l)==0)%SE PONE IF PARA EVITAR DIVISION POR CERO
            x(k,l)=pi/2;
        else
        x(k,l)=atan(M01(k,l)/M10(k,l));
        end
    end
end
```

Anexo F

Código de implementación del método de B-splines:

```
function [CurvaInterpoladaOptima,PuntosControlOptimos] =
MinimizarSnakes(ImaEner,CurvaInterpoladaIni,PuntosControlIni,M,N)

 EnergieMin=Energie(ImaEner,CurvaInterpoladaIni,M);
 PuntosControlMinEner=PuntosControlIni;
 CurvaInterpoladaMinEner=CurvaInterpoladaIni;
 V=1;
 for i=2:N-1

      for dx1=-V:V
      dx1
      x1=PuntosControlIni(i,1)+dx1;

      for dy1=-V:V

      y1=PuntosControlIni(i,2)+dy1;

          for dx2=-V:V
              dx2
               if (i>=1)&(i<=N-1)
               x2=PuntosControlIni(i+1,1)+dx2;

               elseif (i==N)
               dx2=0;
               x2=0;
               end
          for dy2=-V:V
               if (i>=1)&(i<=N-1)
               y2=PuntosControlIni(i+1,2)+dy2;

               elseif (i==N)
               dy2=0;
               y2=0;
               end


              for dx3=-V:V
                  dx3
                 if (i>=1)&(i<=N-2)
                 x3=PuntosControlIni(i+2,1)+dx3;
                 elseif (i>=N-1)

                 dx3=0;
                 x3=0;
                 end
              for dy3=-V:V
                 if (i>=1)&(i<=N-2)
                 y3=PuntosControlIni(i+2,2)+dy3;

                 elseif (i>=N-1)
                 dy3=0;
```

```
                    y3=0;
                    end

                     for dx4=-V:V
                         dx4
                         if (i>=1)&(i<=N-3)
                         x4=PuntosControlIni(i+3,1)+dx4;
                         elseif (i>=N-3)
                         dx4=0;
                         x4=0;
                         end
                      for dy4=-V:V
                         dy4
                         if (i>=1)&(i<=N-3)
                         y4=PuntosControlIni(i+3,2)+dy4;
                         elseif (i>=N-2)
                         dy4=0;
                         y4=0;
                         end



[PuntosControlFin]=MoverPuntoControl(PuntosControlIni,x1,y1,x2,y2,x3,y3
,x4,y4,i,N);


[PuntosControlExt]=ExtenderPuntosControl(PuntosControlFin,N);


                    h=floor(M/N);


CurvaInterpoladaFin=InterpolDeSplines(h,PuntosControlExt,M);
                    CurvaInterpoladaFin=floor(CurvaInterpoladaFin);

[CurvaConexaFin,long3]=CurvaConexa(CurvaInterpoladaFin,M);

                    Ene=Energie(ImaEner,CurvaConexaFin,long3);
                    if(Ene<EnergieMin)
                           PuntosControlMinEner=PuntosControlFin;
                           CurvaInterpoladaMinEner=CurvaInterpoladaFin;
                           EnergieMin=Ene;
                    end
              end
          end
      end
end
end
end
end


end
PuntosControlIni(i,1)=PuntosControlMinEner(i,1);
PuntosControlIni(i,2)=PuntosControlMinEner(i,2);
PuntosControlOptimos(i,1)=PuntosControlMinEner(i,1);
```

```
PuntosControlOptimos(i,2)=PuntosControlMinEner(i,2);

end
CurvaInterpoladaOptima=CurvaInterpoladaMinEner;

function [PuntosControlExt]=ExtenderPuntosControl(PuntosControl,N)
N;
deltax=PuntosControl(N,1)-PuntosControl(1,1);
deltay=PuntosControl(N,2)-PuntosControl(1,2);

%PuntosControlExt(1,1)=PuntosControl(N,1)-deltax;
%PuntosControlExt(1,2)=PuntosControl(N,2)-deltay;
%PuntosControlExt(2,1)=PuntosControl(N-1,1)-deltax;
 PuntosControlExt(2,2)=PuntosControl(N-1,2)-deltay;

PuntosControlExt(1,1)=PuntosControl(1,1);
PuntosControlExt(1,2)=PuntosControl(1,2);
PuntosControlExt(2,1)=PuntosControl(1,1);
PuntosControlExt(2,2)=PuntosControl(1,2);
for i=1:N
PuntosControlExt(i+2,1)=PuntosControl(i,1);
PuntosControlExt(i+2,2)=PuntosControl(i,2);
end

%PuntosControlExt(N+3,1)=PuntosControl(1,1)+deltax;
%PuntosControlExt(N+3,2)=PuntosControl(1,2)+deltay;
%PuntosControlExt(N+4,1)=PuntosControl(2,1)+deltax;
 PuntosControlExt(N+4,2)=PuntosControl(2,2)+deltay;
%PuntosControlExt(N+5,1)=PuntosControl(3,1)+deltax;
%PuntosControlExt(N+5,2)=PuntosControl(3,2)+deltay;

PuntosControlExt(N+3,1)=PuntosControl(N,1);
PuntosControlExt(N+3,2)=PuntosControl(N,2);
PuntosControlExt(N+4,1)=PuntosControl(N,1);
PuntosControlExt(N+4,2)=PuntosControl(N,2);
PuntosControlExt(N+5,1)=PuntosControl(N,1);
PuntosControlExt(N+5,2)=PuntosControl(N,2);


function Ene=Energie(ImaEner,CurvaInterpolada,M)
suma=0;
for i=1:M,
    inx=CurvaInterpolada(i,1);
    iny=CurvaInterpolada(i,2);
    inx=floor(inx);
    iny=floor(iny);
    suma=suma+ImaEner(inx,iny);
end
Ene=suma;


function Betax=FBetaSplines(x)

    if (abs(x)>=2)
    Betax=0;
    elseif (abs(x)>=1)&(abs(x)<2)
    Betax=((2-abs(x))^3)/6;
```

```
    elseif (abs(x)<1)&(abs(x)>=0)
    Betax=2/3 + (abs(x))^3/2 - x^2;
    End


unction [Puntos,long1]=interpoladorlineal(P1,P2);
close all;
clear all;
MaxIter=0;
SeuilVarEnergie=0.01;


% IninicializacionSnake


fid = fopen('seg2g0','r');

for i = 1:512 ,
 ImaEner(i,1:1:512)=fread(fid,[1,512],'char',0);
end
fclose(fid);
imagesc(-ImaEner);
title('Image Filtr Mediana');
axis off;
axis ij;
axis manual;
axis square;
colormap(gray);
colorbar
P1=[200 300];
P2=[90 150];
fid = fopen('grad_g_0.str','r');
for  k=1:512

        ImaEner1(k,1:1:512)=fread(fid,[1, 512],'char',0);

end
for k=1:512

        ImaDir(k,1:1:512)=fread(fid,[1,512],'float',0);
 end
 fclose(fid);
 figure,
imagesc(-ImaEner1);
title('Image Gradient');
axis off;
axis ij;
axis manual;
axis square;
colormap(gray);
colorbar;


fid = fopen('sb0.str','r');

longcurva=length(CurvaInterpolada);
[CurvaConexa,long3]=CurvaConexa(CurvaInterpolada,long1);
```

```
long3

Vin=Energie(ImaEner,CurvaConexa,long3)
NumIter=0;
VarEnergie=100;
CurvaInterpoladaIni=CurvaConexa;
PuntosControlIni=PuntosControl;
N
while NumIter < MaxIter,


   [CurvaInterpoladaFin,
PuntosControlFin]=MinimicerSnakes(ImaEner,CurvaInterpoladaIni,PuntosCon
trol,long1,N);

%plot(CurvaInterpoladaFin(1:1:301,1),CurvaInterpoladaFin(1:1:301,2));

     [CurvaConexa1,long3]=CurvaConexa(CurvaInterpoladaIni,long1);
     [CurvaConexa2,long4]=CurvaConexa(CurvaInterpoladaFin,long1);
     E1=Energie(ImaEner,CurvaConexa1,long3)
     E2=Energie(ImaEner,CurvaConexa2,long4)
     VarEnergie=abs(E1-E2)
     NumIter=NumIter+1

     CurvaInterpoladaIni=CurvaInterpoladaFin;
     PuntosControlIni=PuntosControlFin;


end
end
```