

## ADQUISICIÓN Y PROCESAMIENTO DE IMÁGENES BIOMÉDICAS

### INTRODUCCIÓN A LA ANGIOGRAFÍA

La angiografía es un procedimiento de diagnóstico invasivo que se realiza en una Sala de Hemodinamia, utilizando un angiógrafo. Este equipo es básicamente un tubo de rayos X, que tiene la capacidad de producir imágenes radiológicas de los vasos sanguíneos.



Figura 1: Fotografía de un estudio de angiografía en curso.

El procedimiento para realizar una angiografía comienza con la inserción de un catéter flexible pequeño en una arteria o una vena, previa anestesia local. Posteriormente, se introduce una aguja pequeña en el vaso sanguíneo a través de la cual se coloca un alambre guía. El catéter posteriormente se desliza sobre el alambre y a través de la luz del vaso sanguíneo. Supervisando el catéter en una pantalla de monitor, el operador puede dirigir cuidadosamente la extremidad del catéter a la región de interés.

Una vez en el sitio, se inyecta una sustancia de contraste a través del catéter por medio de un inyector de presión que regula automáticamente el volumen y la velocidad de la inyección. Esta sustancia de contraste llena la luz del vaso sanguíneo y permite que sea radiológicamente visible. Cuando se comienza a irradiar al sector elegido para el estudio, los vasos sanguíneos que contienen la sustancia radio opaca aparecen más oscuros en las imágenes que el resto de las sustancias del organismo.

Además del tubo de rayos X, el equipo cuenta con un reforzador de imagen que recibe la imagen formada y amplificada que se envía a una cámara de vídeo. Hasta hace relativamente poco tiempo, después de finalizado el estudio, el profesional debía retirar los catéteres y esperar el proceso de revelado, fijado, lavado y secado de la película para realizar un diagnóstico certero. Con la aparición de los sistemas digitales, estas demoras ya no existen. Actualmente, la señal de vídeo es posteriormente digitalizada y enviada a un sistema informático para su posterior procesamiento.

Las imágenes digitales obtenidas, tanto de manera individual como agrupadas en conjunto (componiendo un *video* del procedimiento), permiten evaluar con precisión la anatomía arterial y determinar la existencia de estrechamientos (estenosis), obstrucciones, dilataciones anormales o de comunicaciones anormales de los vasos.

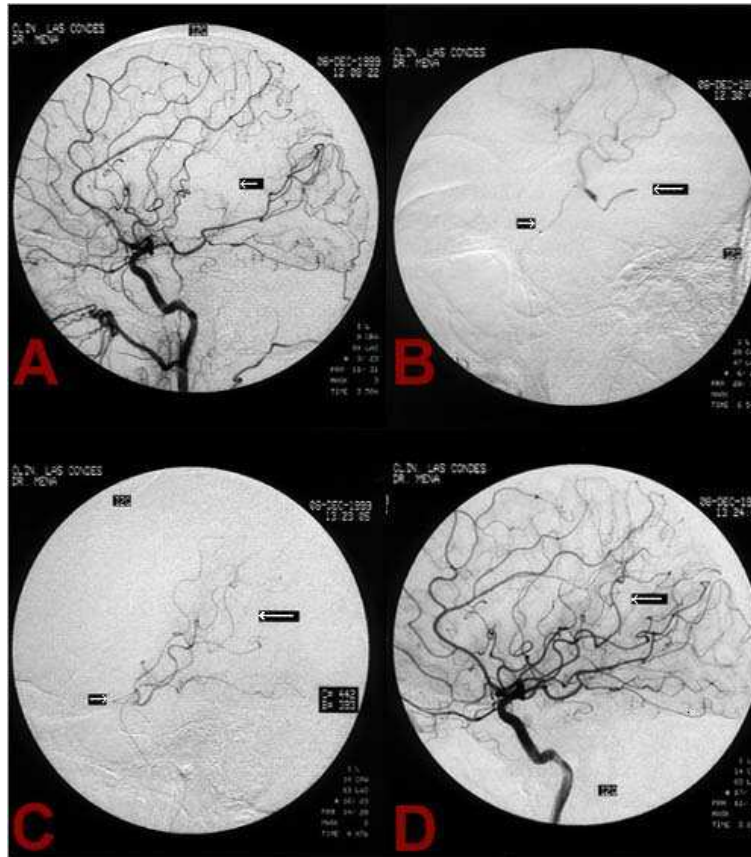


Figura 2: Imagen de un estudio de angiografía cerebral.

Con la angiografía digital, ya no se toman medidas estimativas con papeles apoyados sobre un proyector, sino que esto ha sido reemplazado por un monitor de imágenes, en el cual se tratan las mismas de una manera más sencilla y fiable. Las imágenes pueden ser observadas una a una, detenerse en un punto específico, avanzar o retroceder en el video y aumentar de tamaño o focalizar en un sector determinado.

## 1. ESPECIFICACIONES DEL SISTEMA

Se propone entonces realizar el diseño e implementación de un sistema de adquisición y procesamiento de imágenes angiográficas, que simule el funcionamiento del sistema descrito, con las siguientes características:

- Deberá simular la adquisición de las imágenes mediante un angiógrafo.
- Deberá poseer la capacidad de almacenar estas imágenes en memoria, a fin de permitir la manipulación de las mismas.
- Deberá ser capaz de cargar y guardar imágenes (en formato de archivos) desde o hacia un dispositivo de almacenamiento, respectivamente.

- Deberá poseer la capacidad de procesar dichas imágenes, a fin de mejorar las características y la utilidad de las mismas.
- Deberá poseer la capacidad de mostrar estas imágenes en pantalla, en conjunto con toda otra información que sea relevante para su análisis.

Para mayor detalle, se explicará a continuación en profundidad cada una de las funcionalidades que debe poseer el sistema a fin de cumplir con los objetivos propuestos.

## 2. SIMULACIÓN DE ADQUISICIÓN DE IMÁGENES

El angiógrafo deberá poseer la capacidad de capturar imágenes, pudiendo ajustar además el tamaño de campo de la captura. De esta forma pueden obtenerse imágenes de un sector específico del área de estudio, que puede ser más pequeño que el área máxima que el angiógrafo es capaz de captar. Esto puede ser útil a la hora de focalizar la atención en regiones específicas de la imagen, dejando de lado información posiblemente superflua.

La adquisición de imágenes por parte del angiógrafo será simulada mediante la lectura de archivos desde el disco rígido. Dichos archivos contienen imágenes codificadas en formato *pgm* (*Portable GrayMap*).

Un archivo *pgm* es en esencia un archivo binario, formado por dos partes: una cabecera y los datos de la imagen en sí en formato binario.

La cabecera está codificada mediante caracteres y consta de:

- Un código para identificar el tipo de archivo. Un archivo *pgm* que contiene una imagen de niveles de gris tiene asignado como identificador los dos caracteres "P5".
- Un número indeterminado de comentarios.
- Número de columnas (*c*) y número de filas (*f*), separados por un número indeterminado de espacios.
- Valor del mayor nivel de gris que puede tener la imagen (*m*).

Cada una de estos datos finaliza, normalmente, en un salto de línea.

El contenido de la imagen se almacena como una secuencia binaria de  $c \times f$  bytes, con valores entre 0 y *m*. Cada uno de estos valores representa el nivel de gris de un pixel. El primero referencia al pixel de la esquina superior izquierda, el segundo al que está a su derecha, etc.

Algunas aclaraciones respecto a este formato:

- El número de filas, columnas y mayor nivel de gris se especifica como caracteres (por lo que es posible reconocer los datos de la cabecera abriendo el archivo con un editor de texto).
- Los comentarios comienzan a partir del carácter # y van hasta el final de la línea, pudiendo estar en cualquier parte de la cabecera. Sin embargo, en los archivos de imágenes provistos no hay comentarios en la cabecera.

- Aunque el mayor nivel de gris sea  $m$ , no tiene porqué haber algún píxel con este valor. Éste es un valor que se utiliza mayormente en los programas de visualización de imágenes *pgm*. Para los archivos provistos, el mayor nivel de gris es 255, por lo que el valor de cada píxel puede almacenarse en un dato de tipo **unsigned char**.

Si un archivo con extensión *pgm* es abierto con un editor de texto se puede ver lo siguiente:

```
P5
512 512
255
$' &?67.*,@>01*JZCUQ?XRVhgXOY_LKF]Q
```

Las tres primeras líneas se interpretan con claridad porque fueron grabadas como caracteres, pero a partir de la cuarta fila no puede realizarse una interpretación "entendible" por un editor de texto porque representan números almacenados en formato binario (**unsigned char** – 0 a 255).

### 3. ALGORITMOS DE COMPRESIÓN DE IMÁGENES

Actualmente, la compresión de archivos es una herramienta de gran importancia en la informática, debido en gran medida al aumento continuo en los requerimientos de capacidad de almacenamiento de datos. Existe una gran variedad de algoritmos muy eficientes para este fin, como por ejemplo aquéllos utilizados en los formatos *mp3* en audio y *jpeg* en imágenes.

En este caso, se utilizará un algoritmo genérico, muy simple de implementar. Para esto, se parte del supuesto de que la imagen que se desea comprimir tiene gran cantidad de valores repetidos contiguos, como puede verse, por ejemplo, en las áreas negras que rodean a las imágenes. El algoritmo de compresión consiste entonces en generar un nuevo archivo que contenga el valor y la cantidad de veces que se repite el color.

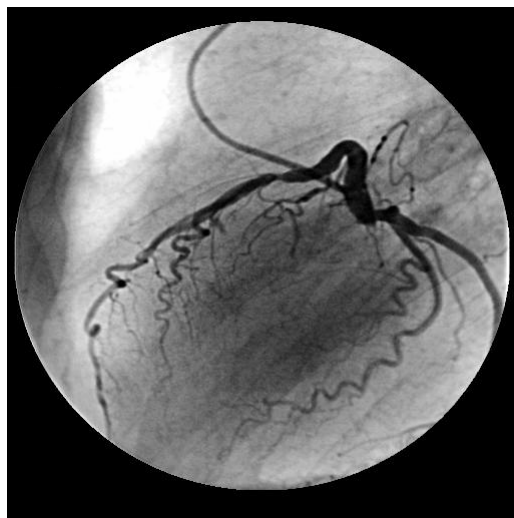


Figura 3: Imagen de un estudio de angiografía coronaria. Nótese la gran cantidad de píxeles negros repetidos en el borde.

Supongamos por ejemplo que los colores de los píxeles del primer renglón de una imagen son los siguientes:

**Secuencia original:** 4 4 4 4 4 4 5 5 5 4 4 4 4 4 4 4 4 4 5 6 6 6 6 7 7 7

La secuencia anterior comienza con el número 4 repetido 6 veces, entonces los dos primeros valores de la secuencia convertida será "4 6" y así sucesivamente.

**Secuencia convertida:** 4 6 5 3 4 9 5 1 6 4 7 3

Como puede observarse, la secuencia convertida contiene la misma información utilizando una menor cantidad de números en el archivo.

Para llevar a cabo esta tarea, deberá contarse con una clase que permita cargar imágenes desde archivos, así como también guardar imágenes en archivos, tanto para las imágenes que están comprimidas como las que no.

#### 4. PROCESAMIENTO DE IMÁGENES – PARTE I

Con frecuencia deben realizarse un conjunto de procesamientos básicos sobre las imágenes obtenidas originalmente, para obtener una imagen clara que pueda ser correctamente interpretable en la clínica. Ejemplos de estos procesamientos son el cambio de brillo y contraste, la ecualización del histograma, la obtención del negativo de la imagen, etc.

En esta primera etapa de procesamiento, las operaciones a implementar consistirán en modificar el brillo de la imagen y obtener su negativo.

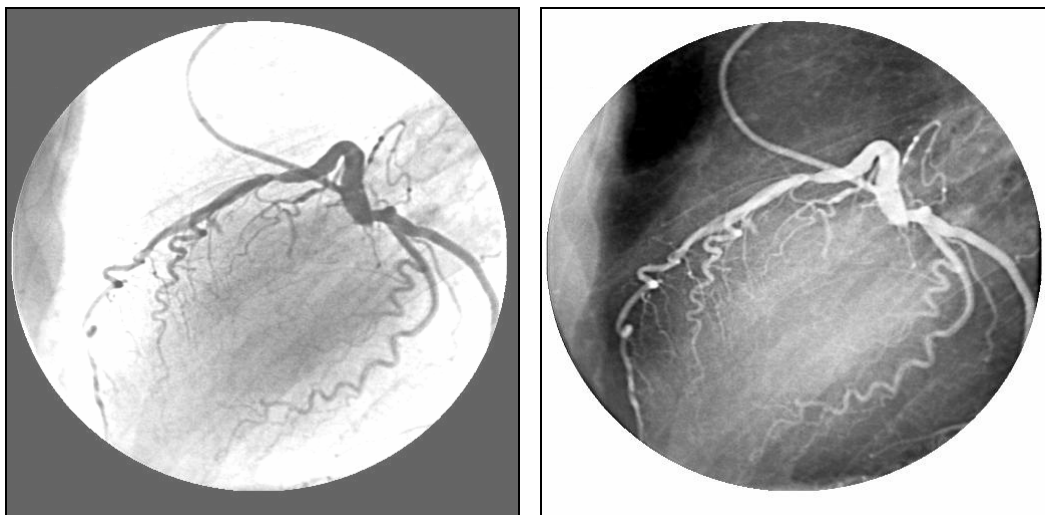


Figura 4: Imágenes de un estudio de angiografía coronaria. Izquierda: aumento de brillo. Derecha: negativo.

Para modificar el brillo de una imagen sencillamente se suma o resta al valor de cada píxel un valor correspondiente a la cantidad de brillo que se desea aumentar o disminuir, respectivamente. Debe tenerse en cuenta que en ningún caso el valor mínimo de un píxel debe ser inferior a 0 ni el valor máximo debe superar a 255.

Para obtener el negativo de una imagen, simplemente se calcula el nuevo valor de cada píxel como la diferencia entre 255 y el valor anterior del píxel.

## 5. PROCESAMIENTO DE IMÁGENES - PARTE II

Una de las herramientas más importantes en el procesamiento de imágenes es la obtención del histograma, que se genera contando la cantidad de píxeles presentes de cada nivel de gris (para estos casos, los niveles de gris varían entre 0 y 255). La forma del histograma permite evidenciar ciertas particularidades de la imagen, como lo son el tipo de fondo, el contraste, y en general si los valores de los niveles de gris están homogéneamente distribuidos o no. Los ejemplos siguientes subrayan estas características:

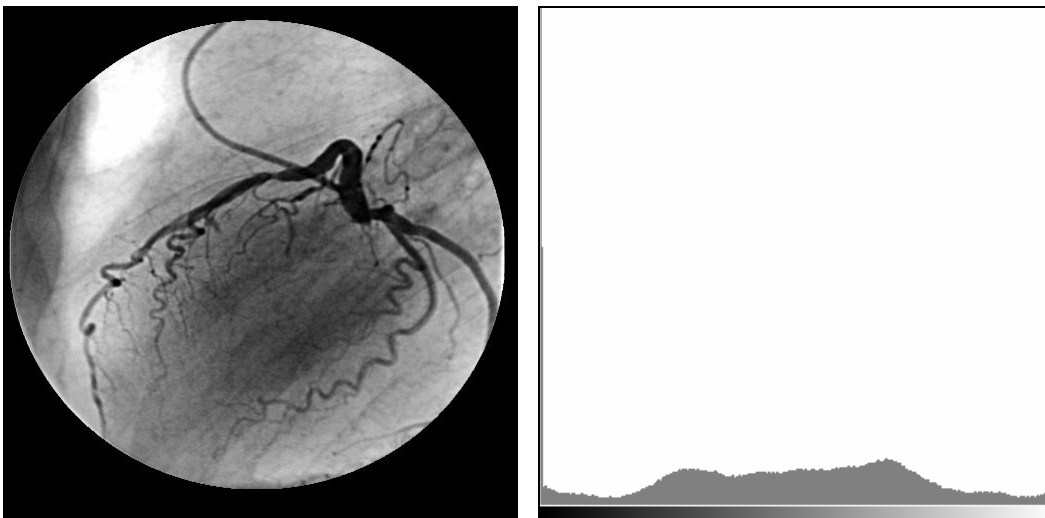


Figura 5: Imágenes de un estudio de angiografía coronaria, con su correspondiente histograma. Nótese la predominancia del color negro (el pico de máxima altura a la izquierda del histograma)

En particular, en muchos casos es deseable que el contraste de la imagen, es decir, la diferencia entre el máximo y el mínimo nivel de gris, sea grande, de forma tal de utilizar todo el rango de intensidades posibles. Un método sencillo (aunque no óptimo) para mejorar el contraste de la imagen es normalizar sus valores, esto es, extender el rango de valores de la imagen de forma tal que el mínimo nivel de gris sea 0 y el máximo nivel sea 255, escalando todos los valores intermedios.

## 6. PROCESAMIENTO DE IMÁGENES - PARTE III

Muchas operaciones de realce de imágenes se hacen en la vecindad de los píxeles o regiones de interés. Esto se debe a que las regiones cercanas al píxel en cuestión pueden proporcionar información valiosa acerca de los niveles de iluminación y los detalles de la escena. El uso de esta información de los píxeles adyacentes está ligado al concepto de *filtrado espacial*.

Una de las técnicas de filtrado espacial consiste en multiplicar una ventana de números o *kernel* por el valor de cada píxel y sus vecinos en una región limitada. Los resultados se suman y el valor final ocupa el lugar del píxel original, y esta operación se repite para todos los píxeles de la imagen.

Este concepto puede explicarse de manera intuitiva como una suma ponderada entre el pixel bajo análisis y sus vecinos. La suma se hace a partir de un grupo de pixeles, o *kernel*, que se encuentra alrededor del (y que incluye al) pixel bajo estudio en su centro. El tamaño del *kernel* es arbitrario, aunque comúnmente se emplean *kernels* de 3x3 a 5x5. La figura siguiente muestra estos dos tipos de vecindad:

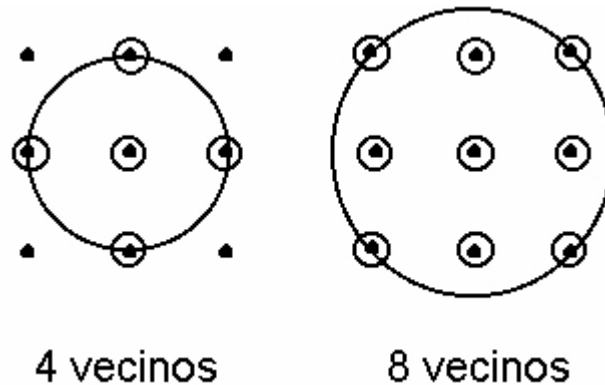


Figura XX: Vecindades alrededor de los pixeles a procesar.

En este tipo de filtrado se realizan modificaciones en las componentes frecuenciales de una imagen. En un filtro de tipo *pasa altos* se efectúa un realce de las componentes de alta frecuencia, de manera que los bordes y los detalles en la imagen se ven más contrastados. En cambio, en un filtro de tipo *pasa bajos* el efecto es diferente, ya que suaviza las transiciones bruscas entre niveles. Dependiendo de los valores con los que se confeccione el *kernel* se puede generar uno u otro filtro.

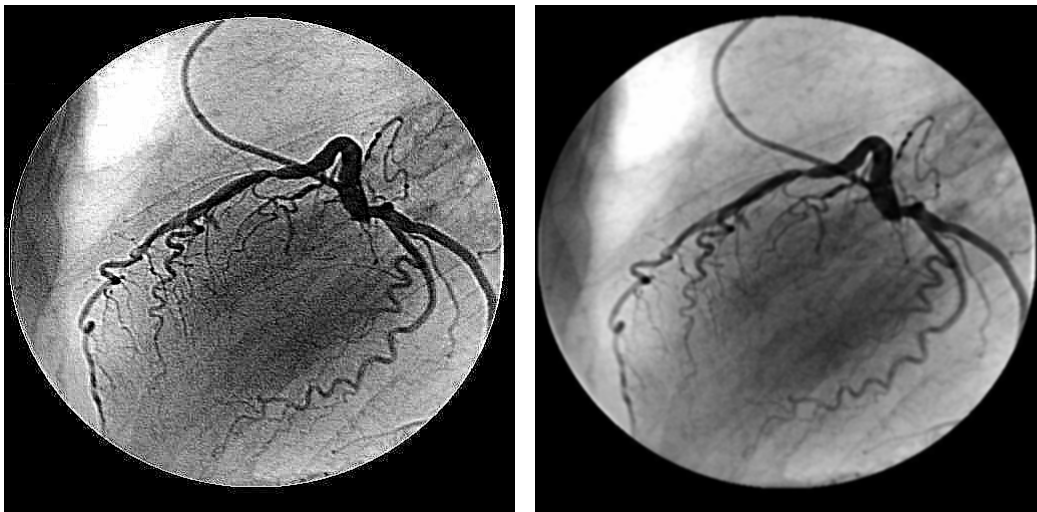


Figura 6: Imágenes de un estudio de angiografía coronaria. Izquierda: luego de un filtrado pasa alto. Derecha: luego de un filtrado pasa bajo.

Además de los anteriores, existen otras técnicas de filtrado, útiles en una gran variedad de casos. Un ejemplo de esto es el *filtro de mediana*. El filtro de mediana es similar al filtrado por *kernels* en el sentido de que cada nuevo valor un pixel que se calcula depende de los valores de los pixeles vecinos. Sin embargo, en este tipo de filtro, el nuevo valor del píxel está

determinado por la *mediana*<sup>1</sup> de los valores de los píxeles vecinos (y no por el promedio, como en el caso de filtrado por *kernel*). La mediana es mucho menos sensible que el promedio a los valores extremos (también llamados *outliers*), por lo que es apropiada para eliminar de las imágenes el ruido compuesto por estos valores extremos (también llamado ruido *sal y pimienta*) sin reducir la nitidez de la imagen.

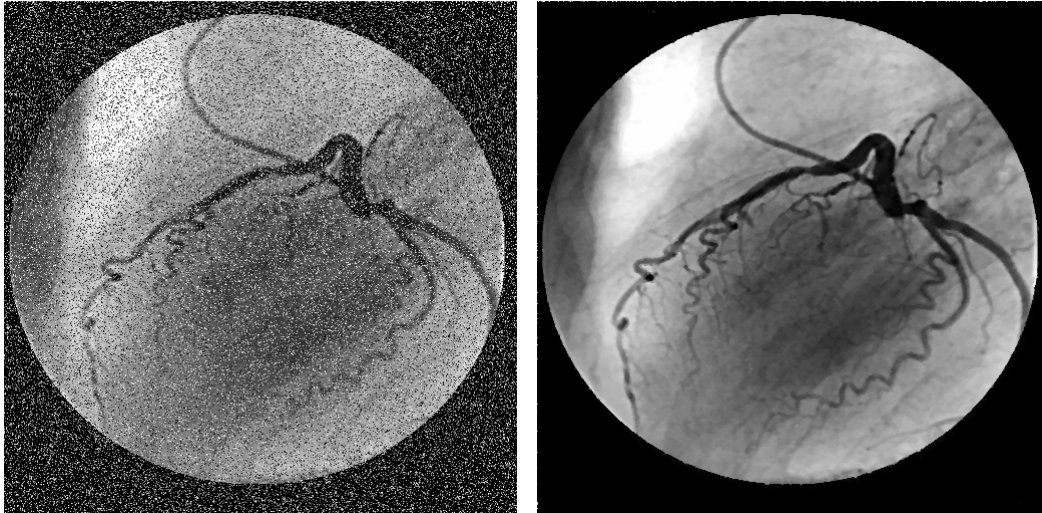


Figura 7: Imágenes de un estudio de angiografía coronaria. Izquierda: con ruido *sal y pimienta*. Derecha: luego de un filtrado por mediana.

## 7. PROCESAMIENTO DE IMÁGENES - PARTE IV

En procesamiento de imágenes, en algunos casos es necesario detectar estructuras específicas. Un ejemplo en este contexto sería la detección de arterias, para determinar parámetros de importancia como pueden ser áreas o formas particulares, que pueden servir para diagnosticar patologías, como estenosis, aneurismas, etc. .

Estas áreas pueden identificarse generando una imagen binaria (conformada sólo por dos niveles de gris, blanco y negro) del mismo tamaño que la imagen original, en la cual los píxeles que definen la región de interés poseen valor máximo (255) y todos los otros píxeles de la imagen poseen valor mínimo (0).

Para generar dicha imagen, es posible utilizar un algoritmo recursivo, denominado *algoritmo del pintor*. El mismo posibilita, a partir de un punto inicial dado, detectar la región de interés compuesta por todos los píxeles con un valor que no difiere del valor del píxel inicial en más de una cantidad determinada.

---

<sup>1</sup> **Mediana:** valor central de una lista ordenada de elementos.



Figura 8: Imágenes de un estudio de angiografía coronaria. Izquierda: original. Derecha: imagen binaria con la morfología de las arterias.

## 8. VISUALIZACIÓN DE IMÁGENES

Por último, se desea contar con un módulo que permita visualizar todas las imágenes generadas, tanto las originales como las procesadas.

También es deseable que dicho módulo permita visualizar los histogramas de dichas imágenes, para brindar al especialista información complementaria a la hora de realizar el análisis.

### TRABAJO PRÁCTICO

#### 1. DISEÑO DEL SISTEMA

- **Objetivo**

Diseñar la solución del problema planteado aplicando Diseño Orientado a Objetos.

- **Trabajo a presentar**

1.a Elaborar el diseño completo del programa: identificación de posibles clases y sus funcionalidades, confección de las tarjetas CRC de cada una de las clases y elaboración de diagramas de jerarquía, agregación y colaboración entre clases, según corresponda.

1.b Realizar la declaración de las clases encontradas.

#### 2. SIMULACIÓN DE ADQUISICIÓN DE IMÁGENES

- **Objetivo**

Aplicar los fundamentos de lectura y escritura de archivos de texto y binarios para la simulación de la captura de imágenes. Utilizar adecuadamente las diferentes estructuras de datos conocidas para almacenar dichas imágenes en memoria.

- **Trabajo a presentar**

2.a Simular la captura de una imagen de las arterias coronarias por parte del angiógrafo. La simulación consistirá en generar una clase que modele el comportamiento de un angiógrafo con la capacidad de leer un archivo con formato *pgm* que contiene la imagen en sí, y posteriormente almacenar esa información en memoria, utilizando para ello una clase que *Imagen* que modele las características de las imágenes.

2.b Simular la captura de una región específica de la imagen completa, cuyo tamaño será determinado por el especialista a cargo del manejo del angiógrafo. Debe tenerse en cuenta que el tamaño de campo a fijar no puede exceder el área máxima de captura del angiógrafo (determinado por el tamaño máximo de la imagen con la que se está trabajando en ese momento).

### 3. ALGORITMOS DE COMPRESIÓN DE IMÁGENES

- **Objetivo**

Aplicar los fundamentos de lectura y escritura de archivos de texto y binarios para la elaboración de algoritmos de compresión de imágenes.

- **Trabajo a presentar**

3.a Generar una clase que permita trabajar con archivos. Esta clase deberá poseer la capacidad de cargar imágenes en la memoria a partir de archivos *pgm* y guardar imágenes a partir de la información almacenada en memoria.

3.b Agregar a dicha clase la posibilidad de comprimir y descomprimir imágenes a elección del usuario del sistema. Para ello debe leerse una imagen de un archivo determinado, realizar la operación (compresión o descompresión) y guardar el resultado en otro archivo. La información de la cabecera no debe ser comprimida y debe ser guardada por igual en todos los archivos (tanto en los que están comprimidos como en los que no). Las operaciones de compresión y descompresión deben ser realizadas de archivo a archivo (no utilizar vectores ni matrices).

### 4. PROCESAMIENTO DE IMÁGENES – PARTE I

- **Objetivo**

Aplicar sobrecarga de operadores para simplificar sintaxis y permitir una lectura más natural del código.

- **Trabajo a presentar**

4.a Sobrecargar los operadores binarios '+' y '-' de la clase que modela imágenes, de manera que permita aumentar o disminuir el brillo de la imagen al sumarle o restarle, respectivamente, un número entero.

4.b Sobrecargar el operador '!' para obtener el negativo de la imagen original.

4.c Sobrecargar el operador '<<' de forma tal de permitir mostrar los valores de los pixeles por la pantalla.

## 5. PROCESAMIENTO DE IMÁGENES - PARTE II

- **Objetivo**

Utilizar de manera eficiente los elementos de la biblioteca STL: contenedores, iteradores y algoritmos.

- **Trabajo a presentar**

5.a Implementar funciones dentro de la clase que modela imágenes que permita calcular y devolver su histograma, utilizando para tal fin el contenedor *map*.

5.b Implementar una función dentro de la misma clase que permita normalizarla, mejorando de esta manera su contraste. A tal fin deberán utilizarse iteradores y algoritmos como *max\_element()* y *min\_element()*.

## 6. PROCESAMIENTO DE IMÁGENES - PARTE III

- **Objetivo**

Aplicar conceptos de polimorfismo para generalizar el comportamiento de los distintos filtros.

- **Trabajo a presentar**

6.a Implementar una clase *Filtro* genérica, de la cual deriven dos tipos específicos de filtro: *Kernel* y *Mediana*.

6.b Implementar los métodos de filtrado de cada clase, tomando para el filtrado por *kernel* la información provista en los archivos provistos por la cátedra.

6.c Implementar funciones para permitirle al usuario seleccionar en tiempo de ejecución con qué filtro trabajar y cuántas veces desea repetir cada acción de filtrado. Los resultados parciales de cada filtrado deben ser almacenados como imágenes por la

## 7. PROCESAMIENTO DE IMÁGENES - PARTE IV

- **Objetivo**

Aplicar conceptos de recursividad para simplificar la codificación de un algoritmo de detección de superficies.

- **Trabajo a presentar**

7.a Implementar una clase encargada de la detección de superficies. Dicha clase debe permitir fijar una región rectangular dentro de la cual buscar la región de interés, fijar el punto inicial y generar una imagen binaria a partir de la detección de dicha región.

7.b Una vez identificada la región, determinar qué porcentaje de la imagen original ocupa.

## **8. VISUALIZACIÓN DE IMÁGENES**

- **Objetivo**

Utilizar las bibliotecas de graficación (OpenGL) y programas específicos de graficación (GNUPlot) para visualizar imágenes en pantalla.

- **Trabajo a presentar**

8.a Implementar una clase que permita visualizar cualquier imagen a elección del usuario mediante el uso de la biblioteca OpenGL, y que permita visualizar el histograma de dicha imagen mediante el programa GNUPlot.