Trabajo Fin de Grado

GRADO EN INGENIERÍA INFORMÁTICA

Facultad de Matemáticas

Universitat de Barcelona

Análisis y optimización de un algoritmo de segmentación de lumen en IVUS: estudio cuantitativo e implementación

Emilio Manzano Marcos

Directores: Simone Balocco y Francesco Ciompi

Realizado en: Departamento de Matemática Aplicada y Análisis. UB

Barcelona, 20 de Junio de 2013

Abstract

Castellano

Las enfermedades coronarias son la primera causa de mortalidad y obesidad en los países desarrollados, por suerte en las últimas décadas se han desarrollado distintas modalidades de extracción de imágenes médicas útiles para el diagnóstico y el seguimiento de pacientes con dichos problemas. Las imágenes IVUS son de las más relevantes en este campo, ya que permiten observar las arterias desde su interior y extraer medidas útiles para el tratamiento de la enfermedad. Con tal de dar apoyo automatizado al médico se está desarrollando un sistema de segmentación automático de lumen, que es la zona por donde circula la sangre en las arterias. En este trabajo trataremos de estudiar su funcionamiento e intentar optimizar este sistema para acercarlo un poco más a una segmentación precisa que sea útil en la práctica médica.

Català

Les malalties coronàries són la primera causa de mortalitat i obesitat als països desenvolupats, per sort en les ultimes dècades s'han desenvolupat diferents modalitats d'extracció d'imatges mediques útils pel diagnostic i el seguiment de pacients amb aquests problemes. Les imatges IVUS són unes de les més rellevants en aquest camp, ja que ens permeten observar les artèries des del seu interior i extreure informació útil pel tractament de la malaltia. Amb la finalitat de donar suport automatitzat al metge s'està desenvolupant un sistema de segmentació automàtic del lumen, que es la zona per on circula la sang a les arteries. En aquest treball tractarem d'estudiar el seu funcionament i intentarem optimitzar aquest sistema per apropar-lo una mica més a una segmentació precisa que sigui útil a la practica medica.

English

Coronary heart disease is the leading cause of mortality and obesity in developed countries, luckily in recent decades have developed various forms of medical imaging extraction useful for diagnosis and monitoring of patients with these problems. IVUS images are the most important in this field since they allow observing the arteries from the inside and take useful measures for the treatment of the disease. To assist doctors work an automatic lumen area segmentation system is being developed, which is the area where the blood flows in the arteries. In this paper we try to study its performance and try to optimize this system to bring it a little more accurate segmentation to be useful in medical practice.

Índice

1. Introducción

En las últimas décadas los avances tecnológicos han facilitado al campo de la medicina la adquisición de imágenes del interior del cuerpo humano, y actualmente dicha información se puede almacenar en forma de imágenes digitales. El estudio de esas imágenes se lleva a cabo utilizando herramientas computacionales, facilitando la adquisición de medidas útiles para el diagnóstico de ciertas enfermedades y su visualización, es el llamado análisis de imágenes médicas.

Existen dos categorías de imágenes: las anatómicas y las funcionales. Las anatómicas representan la morfología, ejemplos de este tipo de imágenes son los Rayos-X, la Tomografía Computerizada (TC), la Resonancia Magnética y los Ultrasonidos. Las funcionales proporcionan información sobre el metabolismo detrás de la morfología, ejemplos son SPECT (Tomografía Computerizada por Emisión de Fotones Individuales) y PET (Tomografía por Emisión de Positrones).

En este proyecto nos centraremos en el desarrollo de métodos de asistencia a médicos para el tratamiento de enfermedades coronarias utilizando Imágenes de Ultrasonidos Intravasculares (IVUS). IVUS es una técnica de adquisición de imágenes basada en ultrasonidos que permite obtener imágenes tomográficas de las estructuras vasculares. Para conseguir ésta herramienta utilizaremos algoritmos de machine learning, reconocimiento de patrones y procesamiento de imágenes que se describirán más adelante.

A continuación se describirán diferentes enfermedades y tratamientos coronarios. Más adelante se expondrá el papel de las IVUS en el diagnóstico, la intervención y la monitorización de pacientes con problemas coronarios. Y finalmente los objetivos de este proyecto y la estructura general de este documento.

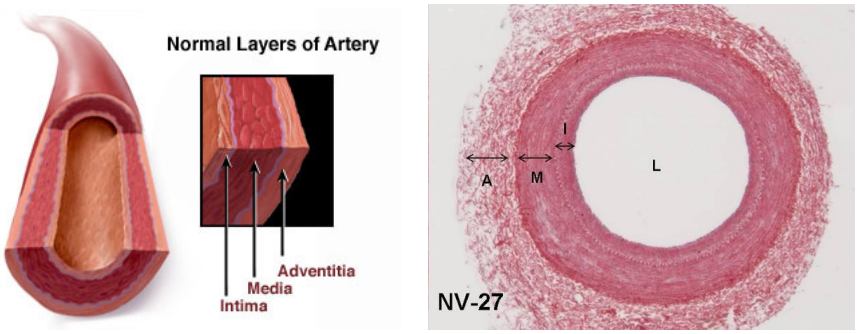
1.1. Contexto médico

Para entender las imágenes y su utilidad primero debemos introducir la estructura de las arterias coronarias y sus enfermedades ateroescleróticas, después analizaremos imágenes IVUS con este tipo de problema médico.

1.1.1. Circulación Coronaria

Las arterias coronarias son vasos que proveen de sangre a los músculos del corazón (miocardio). Son dos: la arteria coronaria derecha y la izquierda.

La arteria coronaria izquierda se divide, casi enseguida de su nacimiento, en arteria descendente anterior y arteria circunfleja. La arteria descendente anterior irriga la cara anterior y lateral del ventrículo izquierdo, y la circunfleja irriga la cara posterior del ventrículo izquierdo. La arteria coronaria derecha irriga fundamentalmente el ventrículo derecho y la región inferior del ventrículo izquierdo.



(a) Capas íntima, media y adventicia.

(b) Histología post-mortem de una arteria coronaria.

Figura 1.1: Estructura de la arteria coronaria

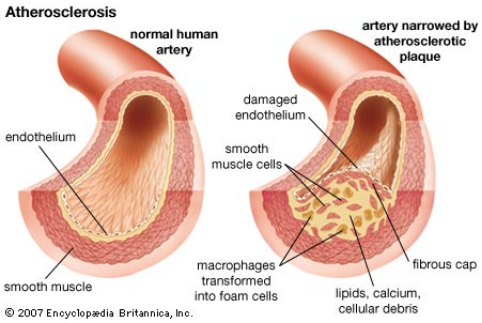
La estructura de las arterias coronarias se compone de tres capas: la mas interna (íntima), la capa muscular (media) y la más externa (adventitia), como podemos observar en la Figura 1.1. La cavidad interior del vaso, por donde circula la sangre es lo que se denomina lumen, como se ilustra en la Figura 1.1b. La capa íntima, en contacto directo con la sangre, está rodeada por el endotelio, una capa compuesta de células epiteliales escamosas simples. El endotelio forma una capa lisa que minimiza la fricción con la sangre. La capa media, que separa la capa íntima de la adventicia, está compuesta por células musculares y tejido elástico, se caracteriza por un bajo contenido en colágeno. Finalmente la adventitia se compone básicamente de colágeno y tejido elástico.

La estructura del árbol coronario está llena de bifurcaciones vasculares, son los lugares donde una arteria se divide en dos vasos hijos: la rama principal y la rama lateral.

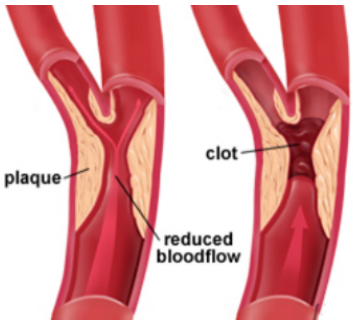
1.1.2. Enfermedades Coronarias

Las arterias coronarias pueden verse afectadas por enfermedades, éstas enfermedades son las principales causas de mortalidad y obesidad en los países desarrollados. Dichas enfermedades son el resultado de un complejo proceso llamado aterosclerosis. Es una enfermedad progresiva y crónica que tiende a la formación de múltiples placas en el interior de las paredes del vaso sanguíneo.

La aterosclerosis empieza con la inflamación, que lleva a la activación endotelial y al reclutamiento de monocitos (glóbulos blancos). Esto daña el endotelio, hecho que facilita la acumulación de colesterol y otros residuos en las paredes internas de la arteria. Los depósitos de lípidos se suelen encontrar en coronarias de tamaño medio-grande, suelen ser puntuales y están distribuidos irregularmente. Éstos depósitos pueden evolucionar en fibrosis (inflamación y engrosamiento) y



(a) Aterosclerosis, arteria sana y con placa.



(b) Las bifurcaciones son zonas de riesgo para la acumulación de placa.

Figura 1.2: Aterosclerosis y bifurcación.

calcificación (depositamiento de calcio). La formación de placa aterosclerótica es el resultado de la proliferación y sucesiva destrucción de tejido de la íntima, lo que lleva a la formación de un ateroma, un engrosamiento de los segmentos de la íntima-media y un engrosamiento general de la pared del vaso.

La aterosclerosis provoca un estrechamiento del lumen, que dificulta el flujo de sangre y oxígeno al corazón y es la causa principal de las enfermedes cardíacas. Además, las placas que contienen un núcleo suave de ateroma son inestables y pueden llegar a romperse, la capa fibrosa que separa el núcleo del lumen puede desintegrarse y provocar una trombosis (Figura 1.2a). Estas placas rotas son las principales responsables de la angina de pecho, el infarto de miocardio, y la muerte súbita.

Las bifurcaciones son puntos críticos desde el punto de vista clínico, ya que son zonas donde aparecen mas frecuentemente las lesiones ateroscleroticas. El estrechamiento de las paredes y la formación de placa en las arterias se suele dar en uno o en los dos vasos hijos de una bifurcación mayor, como podemos ver en la Figura 1.2b. Por otra parte, la ruptura de placa puede depender de parámetros anatómicos como la proximidad a una bifurcación. En particular, la presencia de una bifurcación especialmente después de la lesión significa un riesgo mayor de ruptura de placa y consiguientemente de trombosis.

1.1.3. Intervenciones Coronarias Percutáneas (ICP)

El estrechamiento de la zona de lumen se puede tratar con una intervención. La intervención coronaria percutánea (ICP), conocida comúnmente como angioplastia, es un método no quirúrgico para ensanchar la zona bloqueada y mejorar la circulación de la sangre. Es una alternativa a tratamientos farmacológicos y a la cirugía mayor, y implica un cateterismo cardíaco del paciente. La ICP puede llevarse a cabo durante un diagnóstico (cateterismo) si se identifica una zona

obstruida, o se puede programar para más adelante una vez se ha confirmado la presencia de una obstrucción.

Existen diferentes tipos comunes de intervención: la angioplastia con balón, el despliegue de un stent, la roto-ablación y un balón cortante.

- **Angioplastia con balón:** un pequeño balón en la punta del catéter se posiciona en la zona obstruida de la arteria, una vez ahí el balón se infla y la placa se comprime contra las paredes. Esto aumenta el diámetro de la zona de lumen y permite una mejor circulación sanguínea. Puede ser un método complicado si se aplica de forma recurrente en la misma zona.
- **Despliegue de stent:** en la mayoría de los casos el método anterior se combina con el despliegue de un stent. Un stent es una maya tubular, hecha de un material metálico o bioabsorbible, que actúa reteniendo la contracción de las paredes de la arteria causadas por la placa. Se usa un catéter con balón para insertar el stent en la zona a tratar, una vez está en posición el balón se infla hasta el diámetro deseado, expandiendo el stent y depositándolo para que mantenga la apertura del vaso. Una vez colocado el stent, el balón se desinfla y se retira el catéter.
- **Roto-ablación:** es un catéter especial en forma de bellota con un recubrimiento de diamante. Éste se lleva hasta la zona afectada, y una vez allí, la punta gira a alta velocidad y retira la placa de las paredes de la arteria.
- **Balón cortante:** tiene una punta formada por un balón con cuchillas. Cuando el balón se infla las cuchillas se activan y cortan la placa, despúes el balón comprime la materia grasa de la arteria contra las paredes, como en la angioplastia con balón descrita anteriormente. Éste tipo de intervención se utiliza cuando se ha depositado placa en una zona donde anteriormente se había colocado un stent.

1.2. IVUS

Hasta hace relativamente pocos años, la Angiografía era el único método de adquisición de imágenes durante una intervención. La Angiografía se considera la técnica estrella para adquirir imágenes en enfermedades coronarias, utilizada normalmente para detectar lesiones arteriales, guiar intervenciones percutáneas e intercambiar datos entre médicos. Sin embargo tiene sus limitaciones a la hora de medir la zona de lumen y muchas veces falla en la detección de lesiones propensas a derivar en una trombosis.

En las últimas décadas, las IVUS han evolucionado hasta el punto de ser un valioso suplemento a la Angiografía. Las imágenes IVUS son una modalidad de adquisición de imágenes basada en un catéter que consigue imágenes tomográficas de alta resolución de las estructuras vasculares. El procedimiento para adquirir una secuencia de IVUS consiste en introducir una sonda ultrasónica

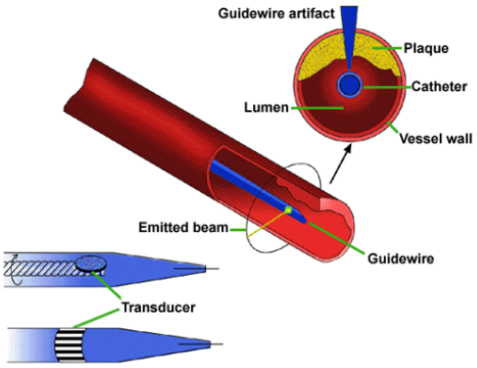


Figura 1.3: Esquema del procedimiento de adquisición de IVUS.

llevada por un catéter en una arteria. El transductor giratorio emite un haz de ultrasonidos mientras se arrastra desde la posición próxima a la distal (pullback) a una velocidad constante, esto se consigue gracias a un motor. El resultado es una secuencia de imágenes. En la Figura 1.3 podemos ver un esquema del proceso.

Un frame IVUS estándar es una vista de una sección transversal de 360 grados de las paredes de un vaso sanguíneo, lo que permite una evaluación precisa de la morfología y la composición del tejido del vaso. También se puede generar una visión longitudinal de la secuencia a lo largo de un diámetro en un ángulo concreto. La vista longitudinal permite aproximar la morfología de una sección del vaso en la orientación que deseamos. La Figura 1.4 muestra tres frames, el primero muestra una bifurcación, el segundo la presencia de placa en la arteria y la tercera la presencia de un stent. También podemos ver una vista longitudinal de la secuencia, en la que la posición de los frames están indicados con líneas verticales.

A diferencia de la Angiografía, que muestra la silueta del lumen, IVUS muestra una perspectiva tomográfica transversal. Esto facilita la medición del tamaño del lumen, incluyendo el diámetro mínimo/máximo y el área de la sección transversal, así como el tamaño del ateroma, la distribución de la placa y su composición.

Es una herramienta importante para la pre y post intervención y el seguimiento de enfermedades coronarias en pacientes. En la práctica se utiliza para analizar las condiciones del vaso, localizar lesiones y decidir el procedimiento a seguir (stent o tratamiento farmacológico). Durante una ICP, se toman secuencias IVUS antes y después de la intervención. Antes de la intervención para localizar la lesión y decidir el tamaño del stent a insertar y después de la intervención para evaluar el resultado, la restauración del flujo sanguíneo y si el stent está bien posicionado. También es útil para monitorear el estado de la

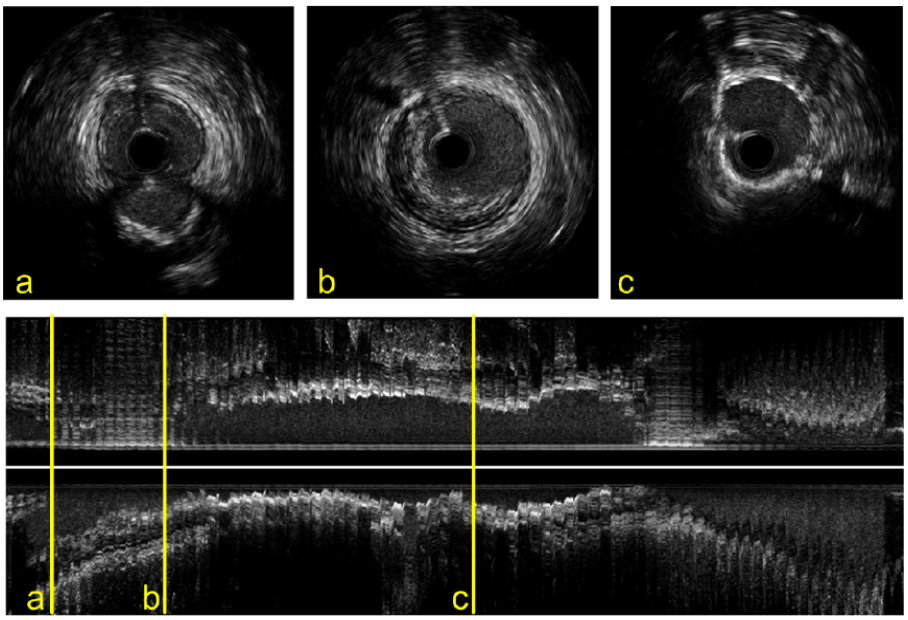


Figura 1.4: Vista de tres secciones transversales mostrando una bifurcación (a), placa (b) y stent (c). Debajo una vista longitudinal de la secuencia con la posición de los frames indicados con líneas amarillas.

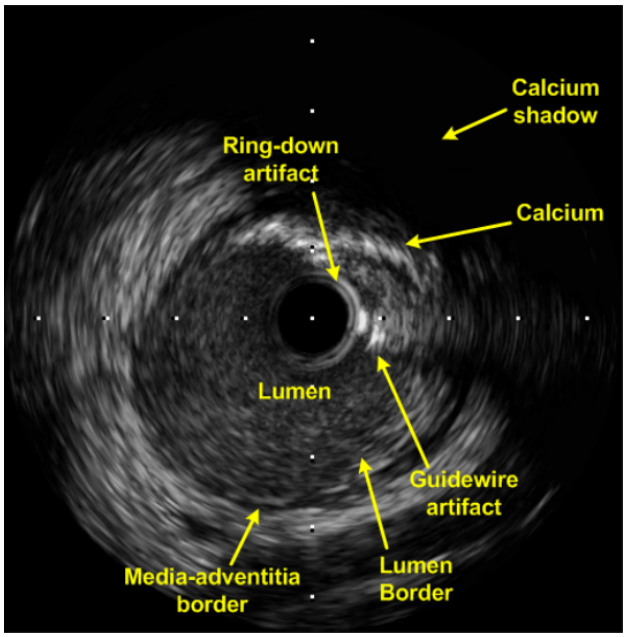


Figura 1.5: Ejemplo de imágen IVUS con diferentes tipos de *artifacts* indicados.

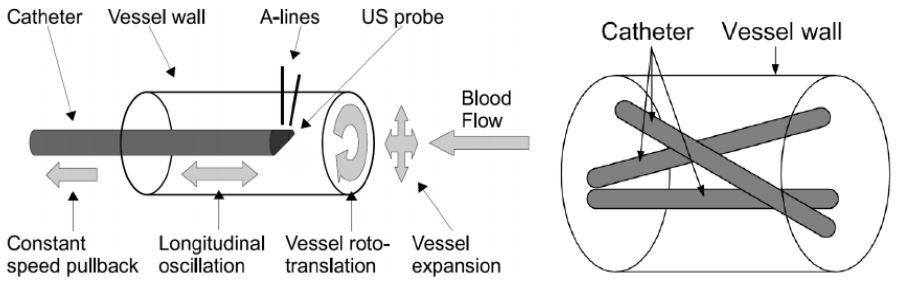
lesión y su evolución. En particular para evaluar la progresión o regresión de la aterosclerosis y la evolución de la composición de la placa.

1.2.1. Dificultades en IVUS

Aunque IVUS es una técnica bastante desarrollada, tiene varios puntos débiles. La aplicación de reconocimiento de patrones y algoritmos de machine learning para analizar automáticamente secuencias de imágenes IVUS a veces es complicada, esto es debido a la aparición de *artifacts*.

La señal ultrasónica emitida por la sonda puede ser reflejada por la guía del catéter y generar ecos y sombras en la imagen. A su vez, el catéter produce una *artifact* circular llamado efecto "ring-down". Las calcificaciones producen sombras que pueden ocultar la morfología del vaso, y también puede surgir ruido de tipo salt & pepper. Algunos de estos problemas se muestran en la Figura 1.5.

Adicionalmente, el movimiento del catéter y el pulso arterial causa *artifacts* dinámicos. El movimiento longitudinal del transductor se ve afectado por un movimiento oscilatorio continuo, esto causa que se tomen varias veces imágenes una misma posición del vaso (ver Figura 1.6a). La posición del catéter no es fija respecto a la morfología del vaso en el plano ortogonal y la rotación de la sonda varía con el pulso del paciente, lo que puede resultar en frames rotados y por lo tanto desalineados. El catéter también puede seguir diferentes trayectorias res-



(a) Esquema de la adquisición de IVUS.

(b) Esquema de las posibles trayectorias del catéter.

Figura 1.6: Adquisición de IVUS.

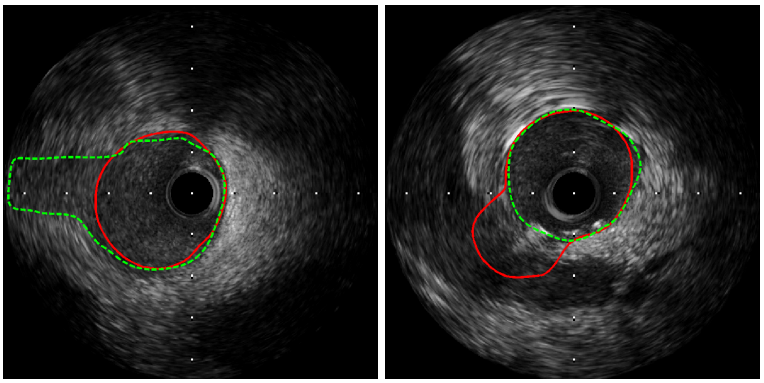
pecto a las paredes del vaso sanguíneo (Figura 1.6b), por lo tanto las secciones que capturamos pueden no ser ortogonales a la pared.

Dados estos problemas, hay ciertas dificultades a la hora de crear una herramienta automática de análisis de imágenes IVUS. La extracción automática de los bordes del vaso, en particular el del lumen y de la media, es muy útil para evaluar el flujo sanguíneo y la cantidad de placa, ya que la placa se localiza entre estos dos bordes. La segmentación automática y semi-automática del borde luminal en IVUS ya ha sido estudiada en profundidad anteriormente, pero es aún un problema abierto debido a la dificultad de desarrollar un método robusto a los diferentes *artifacts*.

1.3. Objetivos

Una vez introducidos los conceptos básicos y las dificultades en la automatización del análisis de IVUS, expondremos cuales son los objetivos iniciales de este proyecto.

Primero de todo hay que saber que partimos de una base, un sistema de detección del borde del lumen que ya da unos resultados bastante buenos. Pero esto no quiere decir que aún quede trabajo por hacer, ya que actualmente el algoritmo, que se explicará en profundidad en la sección 3, tiene ciertos puntos que se pueden y se deberían mejorar. Uno de los problemas principales es la dificultad para detectar las bifurcaciones, esta dificultad ya se ha intentado solucionar anteriormente (Figura 1.7a). También existen algunos problemas en cuanto a la detección errónea de bifurcaciones cuando existe algún tipo de sombra o arteria cercana, esto significa que el método actual en ciertos casos incluye dentro de la zona del lumen zonas que en realidad no le pertenecen (Figura 1.7b). Otro objetivo de este proyecto es el de evaluar si realmente la complejidad que ha adquirido el método es realmente necesaria, es un proyecto que lleva mucho tiempo



(a) El método no detecta la bifurcación.

(b) El método detecta una bifurcación cuando en realidad es una arteria cercana.

Figura 1.7: Ejemplos de errores en el método actual. La segmentación manual en verde, la automática en rojo.

en desarrollo y ha adquirido una alta complejidad computacional, hecho que podría dificultar posteriormente su aplicación en un sistema realmente utilizable para los médicos. El borde que obtenemos actualmente es bastante preciso, por lo que intentaremos mejorar los puntos expuestos sin perder el nivel actual de precisión, de otra forma estaríamos perdiendo parte del trabajo ya realizado.

En cuanto al código del proyecto, al ser un problema tratado por diversas personas y con un largo recorrido en el tiempo, intentaremos refactorizar y estructurar el código para facilitar a posteriores investigadores realizar cambios, reutilizar partes y entender su estructura y uso.

1.4. Motivación

Tenía claro desde hace tiempo que mi trabajo de final de carrera giraría alrededor del campo de la visión artificial. Durante la carrera las asignaturas "Inteligencia Artificial", "Visión Artificial" y en especial la asignatura optativa "Procesamiento de Imágenes" me han despertado cierta curiosidad, me parece un campo realmente interesante y muy útil en infinidad de casos (desde el apoyo automatizado a médicos hasta el reconocimiento de gestos en videoconsolas, pasando por el reconocimiento facial en lugares públicos). Y pensé que con trabajar en un proyecto real de este tipo podría obtener más conocimientos en esta subarea de la inteligencia artificial.

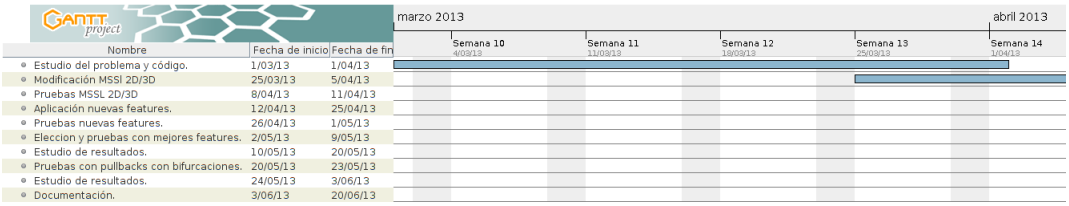
Existe también un motivo personal para trabajar especialmente en las IVUS. Yo estoy operado del corazón, me operaron cuando era muy pequeño, en concreto nací con una estenosis en la válvula pulmonar. Esto es un estrechamiento de

la válvula del corazón por donde sale la sangre desde el ventrículo derecho y entre otras cosas reduce el flujo de sangre hacia los pulmones. Y curiosamente mi operación se llevó a cabo utilizando un catéter y una sonda con balón para ensanchar la válvula. Obviamente como he explicado antes siempre se toman secuencias de imágenes IVUS durante una operación y he tratado de conseguirlas para utilizarlas en el proyecto pero por problemas de tiempo me ha sido imposible, aunque me hubiera gustado poder utilizarlas. Por todo esto tengo especial interés en aportar todo lo que pueda de mis escasos conocimientos a la causa, ya que personalmente me ha ayudado a mejorar mi calidad de vida y espero que pueda ser útil para casos futuros de otras personas.

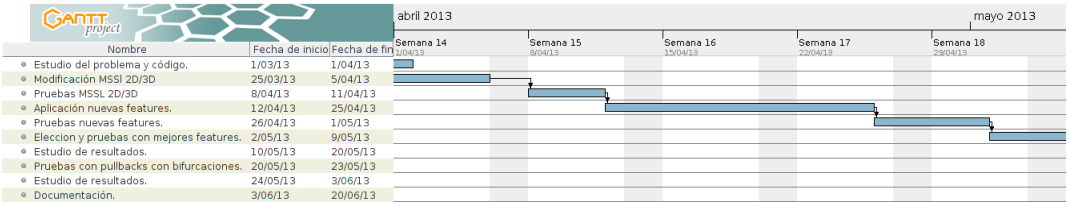
1.5. Planificación

La planificación del proyecto está claramente separada en cuatro partes. La primera parte es un estudio del trabajo ya hecho y sobre el que voy a trabajar, esto incluye estudio de diferentes algoritmos utilizados, familiarización con el tipo de imágenes que voy a usar y comprensión del código del proyecto. Es la parte fundamental para poder decidir el o los caminos a seguir para intentar llegar a los objetivos marcados. La segunda parte es la aplicación de los cambios necesarios, dicho de otra manera, refactorización de código, aplicación de cambios y realización de pruebas y obtención de resultados para poder evaluar posteriormente si suponen alguna mejora. Esta parte es costosa, por lo menos en cuanto a tiempo, ya que depende del tiempo que tarda el clasificador en entrenarse y realizar los tests, que suelen ser horas. La tercera parte es la evaluación y comparación de resultados. La segunda y tercera parte van intercalándose ya que para avanzar e intentar solventar uno de los problemas antes de avanzar debemos cerciorarnos de que lo hecho anteriormente es válido. La última parte engloba todo lo que es la escritura de este documento.

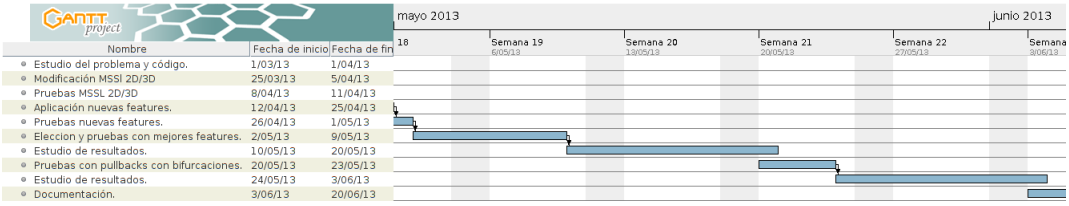
1.5.1. Diagramas de Gantt



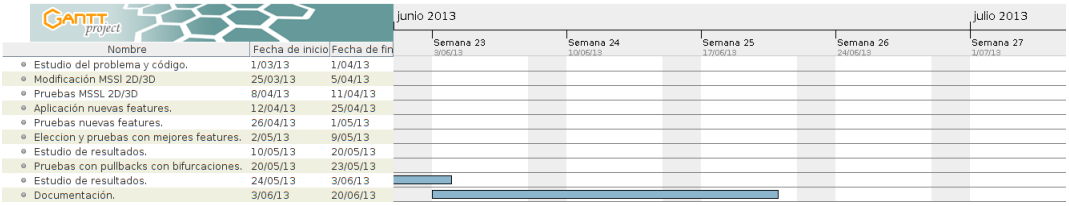
(a) Marzo



(b) Abril



(c) Mayo



(d) Junio

Figura 1.8: Diagramas de Gannt.

1.6. Estructura General de la Memoria

El documento está estructurado en 6 bloques principales.

El primero introduce conceptos necesarios para la comprensión del contexto médico en el que se aplica la automatización de la detección del borde del lumen, también introduce los conceptos de IVUS y los objetivos, motivación y planificación del trabajo. El segundo bloque se centra en las imágenes IVUS más en profundidad, ya que es necesario conocer sus características para poder entender porqué el método automático funciona así.
El tercer bloque expone la estructura y características del algoritmo que utilizaremos, sobre el que realizaremos cambios y sus puntos clave. El cuarto apartado está dedicado al trabajo que he realizado, las modificaciones que se han hecho al sistema, sobre qué medidas realizaremos la evaluación de los resultados y sobretodo el porqué de estos cambios. El quinto bloque (resultados y conclusión) muestra los resultados y las comparativas de los datos (cualitativamente y cuantitativamente), y un resumen de lo que se ha conseguido.
Y la última parte (trabajo futuro) da algunas ideas para posibles mejoras futuras que por falta de tiempo no se han podido realizar.

1.7. Agradecimientos

Debo agradecer a mis dos tutores Simone Balocco y Francesco Ciompi, por su ayuda, sus consejos, su apoyo y su orientación en la realización de este trabajo. También a Petia Ivanova por ser la coordinadora y directora del proyecto IVUS en los últimos 10 años. Finalmente agradecer a Marina Alberti por proporcionarme el código de su tesis sobre el que he podido trabajar.

2. IVUS en Profundidad

2.1. Introducción

En este capítulo se tratan varios métodos de adquisición y análisis automático de imágenes IVUS. En la sección 2.2 se describen las técnicas de adquisición, esto comprende el catéter IVUS, el dispositivo de pullback y la formación de la imágen. La sección 2.2, se explicarán sistemas para compensar los *artifacts* dinámicos (sección 1.2.1) generados por el movimiento del catéter y el pulso cardíaco. Por último la sección 2.4 se presentarán métodos de análisis automático, incluyendo técnicas genéricas de procesamiento de imágenes y reconocimiento de patrones. Algunos de los cuales utilizaremos en éste proyecto, y el estado del arte de métodos para la cuantificación de IVUS.

2.2. Técnicas de Adquisición de Imágenes IVUS

La tecnología IVUS se creó en la mitad de los años 80 y fue probada por primera vez en pacientes humanos a finales de esa misma década. En los años siguientes fue adquiriendo mucha importancia tanto en aplicaciones clínicas como en la investigación.
Las imágenes IVUS se adquieren gracias a las sondas de alta frecuencia, que además son de un solo uso. Estas sondas se introducen en el vaso sanguíneo utilizando un catéter, que avanza por una guía desde la arteria femoral (en la ingle) hasta el lugar de interés de alguna arteria coronaria, para realizar todo esto el médico se apoya en datos de una Angiografía durante el proceso. El catéter tiene una longitud de 150 cm y una punta de un tamaño alrededor de 3.2–3.5 F (1.2–1.5 mm), y permite visualizar segmentos de las coronarias de hasta 15 cm de largo.

En los últimos años han surgido dos tipos de diseño de catéter: un único transductor que rota mecánicamente y otro que contiene un array de transductores a lo largo de la circunferencia. Podemos ver un esquema en la Figura 2.1.

El primer sistema de catéter (mecánico) consiste en un único elemento transductor piezoeléctrico que rota mecánicamente, aproximadamente a unas 1800 rev/min (30 rev/seg). El transductor emite un pulso de ultrasonidos y recibe su señal de retorno, aproximadamente a incrementos de 1^{o} de la rotación angular. El retraso temporal y la amplitud de los pulsos generan 256 escaneos radiales por cada imagen. Éste transductor necesita estar conectado con un cable que pasa por su lado, lo que genera un *artifact* al ponerse en medio del pulso de ultrasonidos de alrededor de 15^{o} de la sección transversal de la imagen. La frecuencia que se suele utilizar en éste tipo de catéter varía entre 12.5 - 40 MHz, aunque algunos llegan a los 45 MHz.
El segundo sistema de catéter (electrónico), posee una placa electrónica que controla un grupo de elementos (transductores) que envían pulsos ultrasónicos sincronizadamente, y a su vez reciben su retorno. Éste sistema circular produce

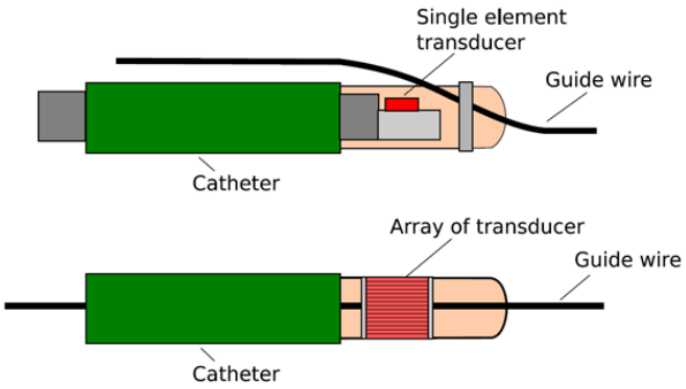


Figura 2.1: Esquema de los dos diseños de catéter: mecánico (arriba) y electrónico (abajo).

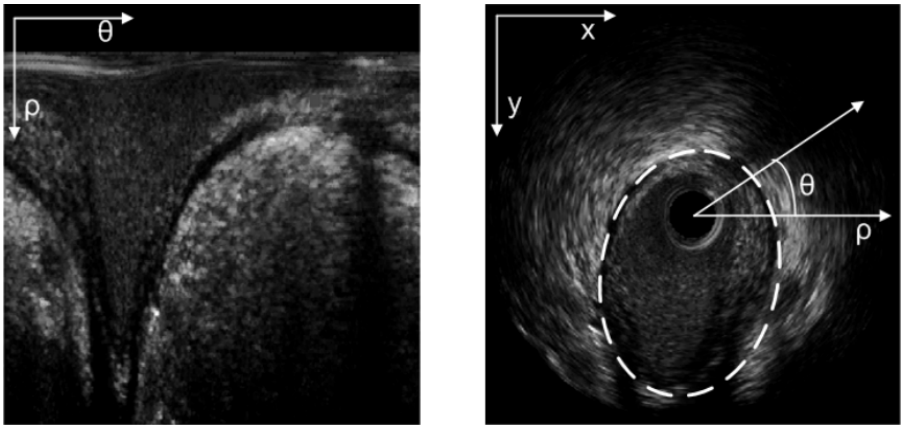


Figura 2.2: Frame IVUS polar (izquierda) y cartesiano (derecha).

imágenes con mayor resolución lateral (resolución perpendicular tanto a la señal como al catéter) que el primer sistema. La frecuencia que se usa oscila entre 15 - 25 MHz.
Con los dos sistemas es posible reconstruir la representación de la sección transversal 360^{o} de la morfología de un vaso. Para formar ésta representación utilizamos las 30 imágenes por segundo que nos proporciona el transductor. Normalmente la resolución típica son 80 micrones axialmente (paralelo al haz de ultrasonidos) y 200 - 250 micrones axialmente (perpendicular al haz y al catéter). El dominio inicial de adquisición es el polar (ρ, Θ) y la imagen resultante a escala de grises es su transformación a coordenadas cartesianas (x, y), como se muestra en la Figura 2.2.

La sonda IVUS va conectada a un dispositivo de pullback y la secuencia de

frames se adquiere conforme se mueve hacia atrás. Hay dos maneras de realizar el pullback del catéter, motorizada y manual. La motorizada tiene ciertas ventajas sobre la manual, la retirada del catéter es constante, lo que previene de cometer una retirada demasiado rápida o demasiado lenta, y permite al médico centrarse en observar las imágenes ya que no tiene que concentrarse en la manipulación del catéter. Éste método motorizado permite mediciones de longitud y volumétricas sobre los datos, así como una adquisición uniforme de las imágenes. Por contra, es más difícil examinar zonas de interés ya que no podemos mantener en un mismo punto el catéter durante el tiempo que queramos. En cuanto a la velocidad de movimiento, muchos expertos aconsejan una velocidad de 0.5 mm/seg.

2.3. Preprocesamiento de Pullbacks: Gating

Como se ha comentado anteriormente, durante la adquisición de secuencias IVUS, el catéter se ve afectado por *artifacts* dinámicos producidos por el latido del corazón y por la posición del mismo catéter. Esto puede interferir con la visualización, la interpretación y el análisis de las imágenes, pero puede ser compensado utilizando técnicas de procesamiento de imágenes.

- **Gating basado en imágen:** El *artifact* más relevante es el causado por el pulso cardíaco, que genera repetidamente una oscilación a lo largo del eje del vaso (conocido como efecto de balanceado), resultando en un posible muestreo múltiple de la misma posición de la arteria. El pullback presenta una oscilación en la dirección longitudinal que se puede ver fácilmente en el vídeo de un pullback motorizado.
  Con el fin de obtener una reconstrucción única de las secciones transversales del vaso, una posible solución es seleccionar los frames que pertenecen a una misma fase del ciclo cardíaco, ya que tendrían una misma rotación. Esta tarea se lleva a cabo utilizando una técnica de "Gating", que tiene como objetivo extraer frames estables y equidistantes. Con estable nos referimos a que una vez hecho el gating, los frames cercanos entre sí del vaso, deben tener una posición y rotación similares.

- **Registro para la Reducción de Oscilación de eje corto:** El segundo *artifact* más relevante es la oscilación y deformación de eje corto. Las variaciones en la posición del catéter con respecto al centro del vaso y la torsión del catéter durante el pullback causa desalineamientos en los frames consecutivos de la secuencia. El centro del vaso no suele estar alineado con el centro de la imagen y el eje de la arteria sufre traslaciones a lo largo de la secuencia. Además, la rotación periódica generada por el pulso cardíaco, puede causar que algunas estructuras importantes (como la placa) aparezcan y desaparezcan a lo largo de la secuencia. Con la intención de minimizar este efecto, se debe alinear el centro del vaso en las imágenes sucesivas y compensarlas por la roto-traslación del pullback.

Existen métodos automáticos para solventar estos problemas pero no son una parte fundamental en este trabajo, así que no vamos a profundizar en cómo

funcionan internamente.

2.4. Métodos de Análisis Automático de IVUS

En este apartado vamos a enumerar diversas técnicas que son parte del estado del arte del análisis automático de IVUS, estas técnicas forman parte de métodos de procesamiento de imágenes, aprendizaje automático y reconocimiento de patrones. Las técnicas que se utilizan para la extracción automática del borde del lumen se explicarán más detalladamente en la siguiente sección.

Procesamiento de Imagenes

- **Filtros:** el filtrado es una clase de procesamiento de señal, su finalizad es la supresión total o parcial de algún componente o característica de la señal. En procesamiento de imágenes se suelen utilizar para extraer características (features) de interés de una imagen.
- **Patrones Binarios Locales:** son operadores de textura, etiquetan un píxel de una imagen haciendo un thresholding (por ejemplo, considerando la intensidad de gris del píxel central como threshold) de los píxeles vecinos y considerando el resultado como un número binario. Los PBL se usan para detectar texturas uniformes en vecindades circulares. En la Figura 2.3 podemos ver un ejemplo.

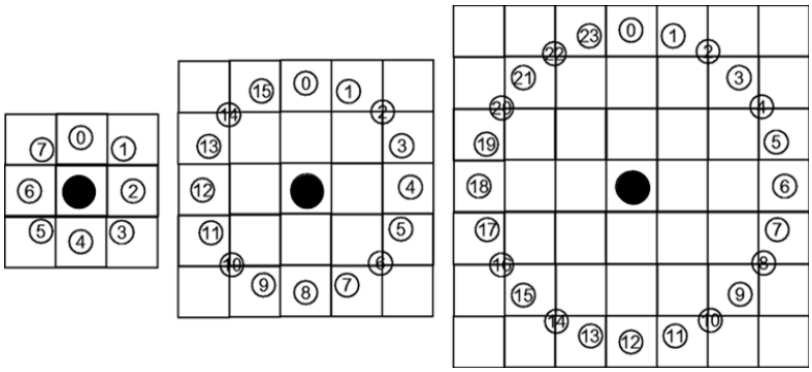


Figura 2.3: Tres vecindades de un píxel central utilizados en PBL.

- **Filtros de Gabor:** los filtros de Gabor son funciones gaussianas moduladas por una sinusoide. Son filtros lineales que se utilizan para la extracción de características o features de textura de una imágen en una orientación en particular. En la Figura 2.4 podemos ver una representación gráfica de un filtro de Gabor.

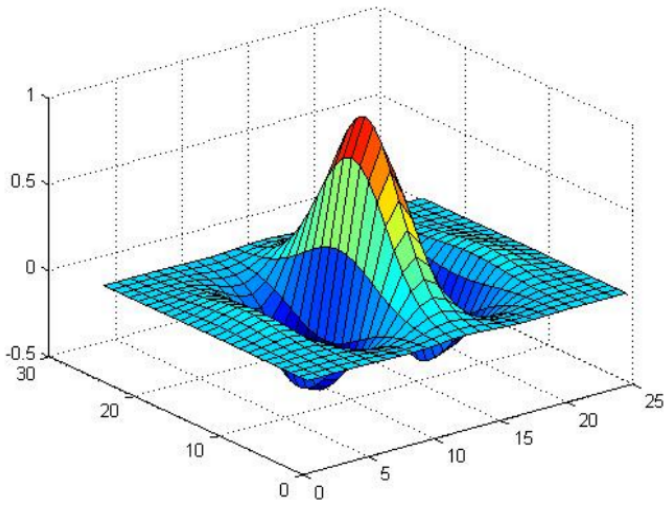


Figura 2.4: Representación gráfica de un filtro de Gabor.

- **Filtrado morfológico:** es un filtrado utilizado para detectar estructuras geométricas predefinidas en una imagen, el resultado es una imagen binaria.
- **Active Contour Models (snakes):** generan un contorno guiados por fuerzas de restricción internas y la influencia de fuerzas externas (en un mapa binario) minimizando la energía, se suelen utilizar en la segmentación de imágenes.

Técnicas de reconocimiento de patrones

En estas técnicas se entrena un clasificador a partir de muestras previamente etiquetadas. Después el clasificador se encarga de discriminar y decide a que clase pertenece cada elemento del conjunto de test. Es un algoritmo de aprendizaje automático y en el caso de las imágenes utiliza información extraída con los filtros mencionados anteriormente.

Clasificadores

- **Clasificador débil y fuerte:** un clasificador débil es un clasificador que se basa en una regla simple, por ejemplo, en una imagen: si es azul es agua, si es verde es hierba. Un clasificador fuerte combina múltiples clasificadores débiles en uno único, mejorando mucho su eficacia.
- **AdaBoost:** es un algoritmo de clasificación que genera un clasificador fuerte (*strong classifier*) a partir de la combinación de varios clasificadores débiles (*weak classifiers*), asignándoles un peso a cada uno.

- **Support Vector Machine:** es un clasificador binario que genera un hiperplano N-dimensional que separa las muestras en dos categorías. En el caso más simple el hiperplano es una linea. Se considera un clasificador eficiente, pero si la cardinalidad del set de training es es grande, normalmente requiere una alta cantidad de memoria y sufre de la necesidad de una alta capacidad computacional.
- **Random Forest:** es un clasificador basado en un árbol de decisión. Es capaz de manejar un número muy grande de características o features robustamente.
- **Multiscale Stacked Secuential Learning (MSSL):** es un meta-clasificador contextual, se encarga de mejorar el resultado de un primer clasificador. Utiliza la correlación espacial entre píxeles a diferentes escalas para decidir a qué clase pertenece cada píxel de un mapa de probabilidades [1].

3. Extracción Automática del Borde del Lumen

3.1. Introducción

En esta sección explicaremos cómo funciona el método de extracción de lumen del que partimos, así como sus características y su estructura general.

Nos centraremos en exponer los métodos mencionados en la sección anterior que se utilicen en nuestro sistema de segmentación IVUS, y profundizaremos en su funcionamiento y su papel en nuestro clasificador. También veremos qué tipo de datos de entrada utilizamos y qué datos de salida genera el algoritmo.

3.2. Características y funcionamiento general

La meta del clasificador es generar una segmentación lo más parecida posible a la hecha manualmente por médicos, teniendo en cuenta todas las dificultades que supone trabajar con imágenes IVUS (sección 1.2.1). Conforme se ha ido estudiando el problema de la extracción del borde del lumen, se han ido probando distintos algoritmos para intentar mejorar los resultados. Se fueron descartando los que daban peores resultados y manteniendo los que producían una segmentación más precisa, hasta llegar a la configuración actual.

Primero explicaremos como funciona la fase de training de este clasificador.
Partimos entonces de varios conjuntos de frames de pullbacks de distintos pacientes, frames que se han extraído utilizando las herramientas de preprocesamiento de pullbacks (sección 2.3). Con esto queremos decir que no utilizamos absolutamente toda la secuencia de imágenes, sino los frames alineados, sucesivos y únicos de una secuencia, a estos frames los denominaremos a partir de ahora "frames gated". En concreto tenemos ocho pullbacks de distintos pacientes, cada uno con sus peculiaridades para abarcar todas las posibles dificultades (presencia de stent, bifurcaciones y sombras), y con distintos niveles de aterosclerosis. Éstas imágenes tienen un tamaño de 512 x 512 píxeles, y por cuestiones de extracción de características que mencionaremos más adelante por cada frame también utilizamos la imagen inmediatamente anterior y posterior (no gated).
Por cada frame necesitamos también un archivo que contiene los *labels*, es decir, las coordenadas de la imagen que forman el borde dibujado por el médico (*ground truth*). Todos estos datos conforman los datos de entrada, y cabe mencionar que son datos en el dominio de las coordenadas cartesianas aunque el sistema trabaja en el dominio polar, por lo que éstos datos se convierten de coordenadas cartesianas a coordenadas polares (Figura 2.2).

A partir de aquí entran en juego los distintos algoritmos y filtros para la extracción de características y clasificación.
El primer paso es la extracción de características de todos los frames y su almacenaje para su uso posterior en el entrenamiento y test del clasificador. Las características utilizadas hasta el momento son ocho, que se detallarán en la

sección 3.3.1, esto genera un vector de ocho características por cada píxel de la imagen. Teniendo en cuenta que después de la conversión las imágenes polares tienen un tamaño de 228 x 256 px, por cada imagen obtenemos una matriz de tamaño 58368 x 8 donde las filas son los píxeles y las columnas son las features. Éste es un paso con un gasto de tiempo considerable (unos 30 minutos) ya que se hace para cada frame de cada pullback.

Una vez tenemos las features para todos los píxeles de todas las imágenes de entrada y los *labels* de todas las imágenes ya podemos entrenar el clasificador. Hay que explicitar que el número de frames que utilizamos para entrenar el clasificador puede variar por problemas de memoria, en mi caso he utilizado 30 imágenes por pullback ya que al almacenarse una cantidad de información muy grande en memoria RAM me ha sido imposible usar más de 30 frames, en otros equipos ese número se podría incrementar ligeramente. El equipo que he utilizado es un Intel Core i7 con 6 Gb de RAM, un equipo con más memoria podría tratar más frames a la vez. Lo que nos deja con un conjunto de training de 30 imágenes x 8 pullbacks = 240 frames. De este conjunto de training se escogen aleatoriamente un subconjunto de muestras (píxeles) para entrenar el clasificador, en concreto 10.000 de cada clase (lumen y no lumen), en total 20.000 muestras.
El clasificador consta de dos fases, son dos clasificadores en cascada, e intenta clasificar los píxeles de la imagen en dos clases (lumen o no lumen), por lo tanto es un clasificador binario. El algoritmo que implementa es una versión MSSL extendida a 3 dimensiones. El primer estadio del MSSL utiliza AdaBoost (sección 3.3.2), y el segundo estadio una versión MSSL 3D (sección 3.3.3) desarrollada por Marina Alberti en su tesis [3]. AdaBoost es un algoritmo de aprendizaje automático que a partir de la combinación diversos *weak classifiers* genera un clasificador fuerte o *strong classifier*. Por lo tanto en esta primera fase se genera un clasificador fuerte a partir de las ocho features iniciales y los *labels* creados por el médico.
Para el segundo estadio, primero debemos generar la probabilidad de pertenecer a una de las dos clases para cada píxel de todas las imágenes del conjunto de training, a partir del primer clasificador, ya que MSSL utiliza esos datos para crear un conjunto de características contextuales espaciales. Por lo tanto se aplica el primer clasificador al conjunto de training, se extraen las features contextuales con MSSL, se añaden a las features iniciales y se vuelve a aplicar AdaBoost para crear un segundo clasificador fuerte.
El resultado de este proceso son dos clasificadores fuertes que podemos utilizar para generar un mapa binario (lumen vs no lumen) en un conjunto de test (Figuras 3.1 y 3.2).

En cuanto a la fase de test, la primera fase extrae las features de un conjunto de frames de test, aplica el primer clasificador y crea un mapa de probabilidad de cada imagen. En la segunda fase se extraen las características contextuales con MSSL a partir de los mapas anteriores, y finalmente se clasifican los frames con el segundo clasificador creando mapas binarios de clasificación finales. Una

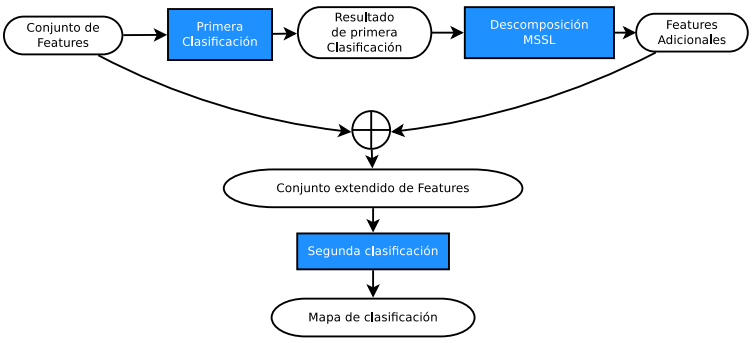


Figura 3.1: Diagrama del proceso de test.

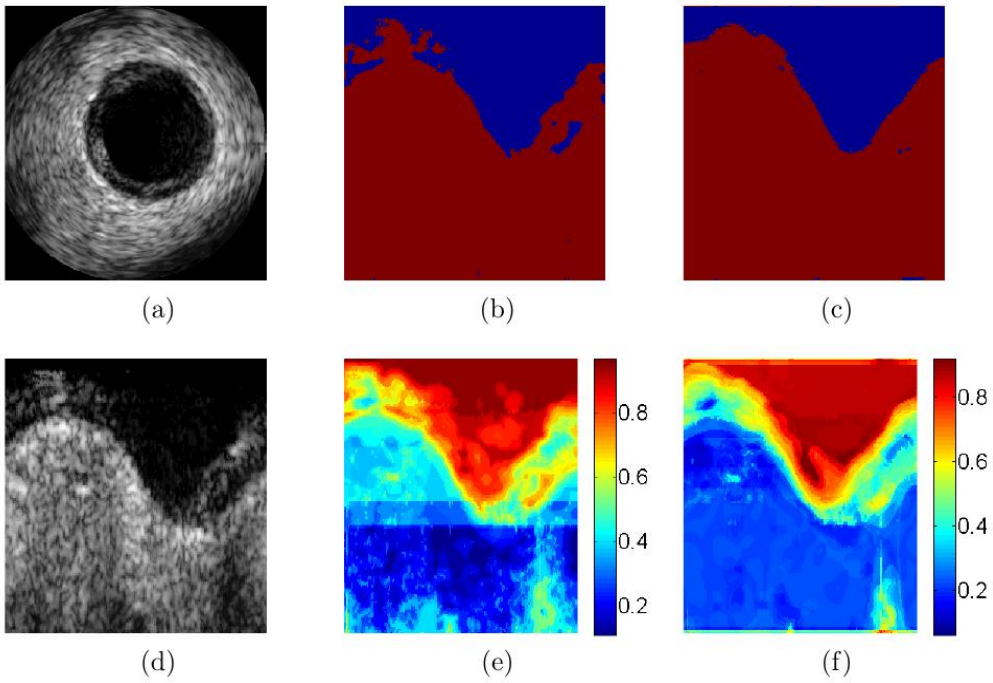


Figura 3.2: Frame IVUS en versión cartesiana (a) y polar (d). La imagen b muestra el mapa binario de la salida del primer clasificador, y la imagen c el mapa binario generado por el segundo clasificador (MSSL 3D). Las imágenes e y f, muestran el mapa de pseudo-probabilidad de los dos clasificadores respectivamente.

vez tenemos dichos mapas se aplica un algoritmo *Active Contour Model (Snake)*[7]. Éste algoritmo se encarga de generar un borde que se ajuste al borde del mapa binario, para así tener definida la segmentación final de la imagen. En la Figura 3.3 podemos ver un diagrama del funcionamiento general del método (training y test).

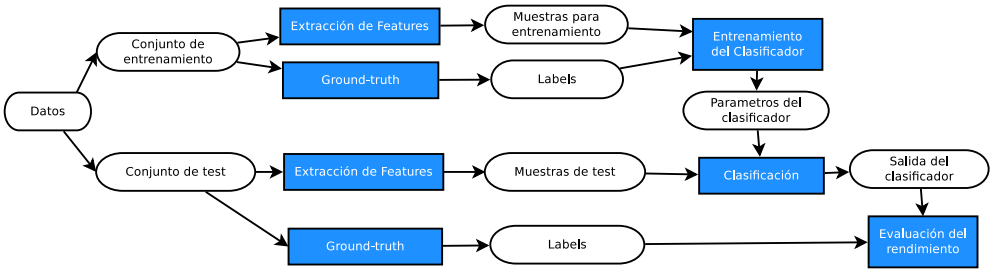


Figura 3.3: Diagrama general del funcionamiento del sistema de segmentación.

3.3. Clasificador

A continuación detallaremos los puntos más relevantes del clasificador.

3.3.1. Features

Las características que se extraen de las imágenes son un punto clave en el desempeño del clasificador, por ese motivo es importante escoger features que nos den una información relevante de la imagen. En el método de segmentación actual se utilizan ocho features, el número de características es importante porque puede aumentar considerablemente la complejidad y la eficiencia del método, y reducir el tiempo computacional es crucial para que sea utilizable en la práctica clínica.

Estos son los filtros que aplicamos para la extracción de características:

- **Nivel de gris:** Es la imagen original, se utiliza el nivel de gris de cada píxel como característica. Las imágenes que mostraremos en las siguientes figuras serán ésta imagen original (Figura 3.4) filtrada con el filtro correspondiente.

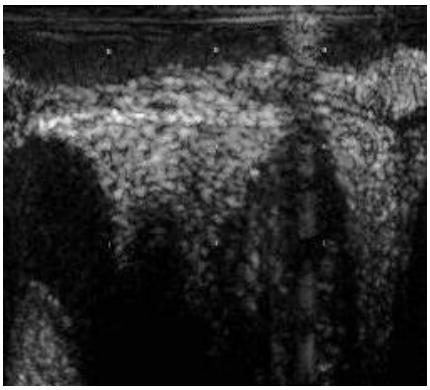


Figura 3.4: Imagen original, el nivel de gris se utiliza como feature.

- **Filtros de Gabor**: En el proyecto se utilizan cuatro filtros de Gabor. Son filtros lineales formados por la multiplicación de una función sinusoidal y una función gaussiana. La ventaja de estos filtros es que trabajan tanto en el dominio espacial como frecuencial al modularse con una función gaussiana, al contrario que las funciones sinusoidales que trabajan sólo en el domino frecuencial. El filtrado de las imágenes con funciones de Gabor está relacionado con los procesos que se llevan a cabo en la corteza visual, son un modelo para los campos receptivos de las células de la corteza cerebral, y se ha demostrado que tienen una buena respuesta en tareas de segmentación de texturas.
  Los cuatro filtros utilizados se diferencian en la orientación, cada uno utiliza un ángulo diferente. En la Figura 3.5 podemos ver un ejemplo de una imagen filtrada con un filtro de Gabor.

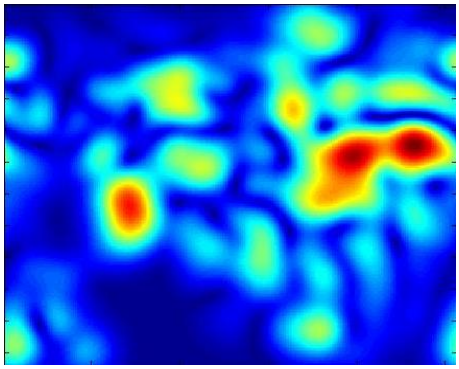


Figura 3.5: Imagen filtrada con Gabor (ángulo 0).

- **Shade:** Este filtro trabaja verticalmente, columna a columna, ya que es-

tamos trabajando en imágenes polares y la zona de lumen siempre va a estar en la parte superior de la imagen. Se basa en la acumulación de nivel de gris. A cada píxel de la imagen le asigna el valor medio de todos los píxeles situados debajo de él verticalmente. Podemos ver la función del filtro, dónde BI es una imagen binaria obtenida haciendo un *threshold* TH de la imagen original y N_r y N_c es el tamaño de la imagen. El valor de TH se ha fijado experimentalmente en 14 [2].

$$Sh\,(x,y) = \frac{1}{N_r N_c} \sum_{y_s=y}^{N_r} BI\,(x,y_s) \tag{1}$$

En la Figura 3.6 podemos ver la salida de este filtro.

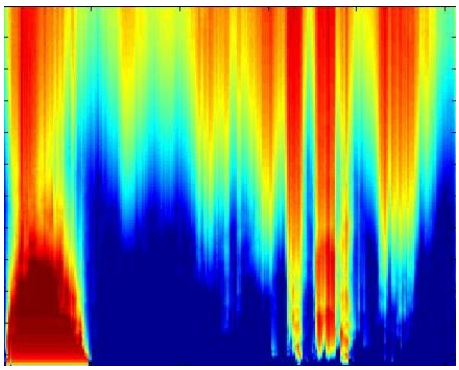


Figura 3.6: Imagen filtrada con Shade.

- **Relative Shade:** Trabaja igual que el filtro Shade, con una diferencia, en cada píxel multiplica el valor obtenido de la media de los píxeles inferiores (como en Shade) por la posición vertical del píxel. De esta manera damos un peso distinto al valor dependiendo de la posición vertical del píxel, un valor relativo [2]. La función Relative Shade:

$$Sh\,(x,y) = \frac{1}{N_r N_c} \sum_{y_s=y}^{N_r} y_s BI\,(x,y_s) \tag{2}$$

La salida de este filtro visualmente es idéntica a la del filtro Shade (Figura 3.7), aunque los valores en cada píxel son distintos.

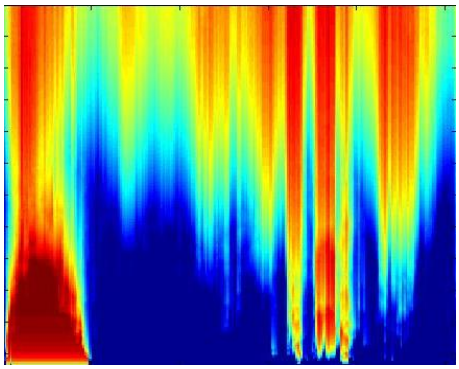


Figura 3.7: Imagen filtrada con Relative Shade.

- **Correlación de Pearson:** Aplica el coeficiente de correlación de Pearson sobre una ventana de tamaño 14 x 14 px en toda la imagen.
  Esta correlación se hace sobre los frames inmediatamente anterior y posterior de la imagen, es por eso que en el conjunto de imágenes de entrada necesitamos, además de los frames gated, esos dos frames que se extraen del pullback original completo. La correlación se hace entre los 3 frames, primero con segundo, segundo con tercero y primero con tercero, y la media de estos tres resultados se combinan en un único valor para cada píxel.

  El coeficiente de correlación de Pearson es un índice que utiliza para medir el grado de relación entre dos variables a partir de la siguiente fórmula, donde "x" e "y" son conjuntos de valores:

$$r_{x,y} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2} - \sqrt{n\sum y_i^2 - (\sum y_i)^2}} \tag{3}$$

  Este filtro se basa en la premisa de que la sangre que fluye rápidamente por la zona de lumen produce cambios significativos del patrón de ruido "speckle" entre dos frames adyacentes en esa zona, mientras que las demás partes de la imagen (placa y paredes del vaso) no varían significativamente. El resultado de este filtro se puede apreciar en la Figura 3.8.

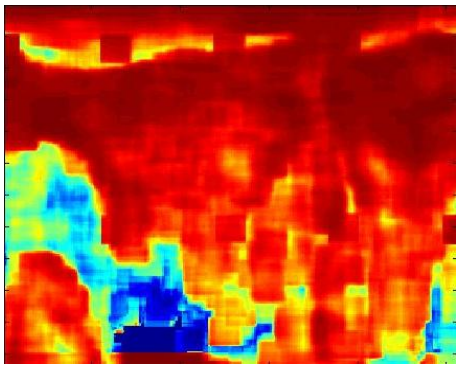


Figura 3.8: Imagen filtrada con la correlación de Pearson.

3.3.2. AdaBoost

El corazón de nuestro sistema de detección del borde de lumen es AdaBoost. A grandes rasgos es un algoritmo de aprendizaje automático que nos permite generar un clasificador fuerte a partir de múltiples clasificadores débiles.

Para entender cómo funciona AdaBoost, primero debemos entender los conceptos de *Bagging y Boosting.*

Bagging

La técnica de *Bagging* (*bootstrap aggregation*) consiste en la generación de T hipótesis h_t a partir de subconjuntos de entrenamiento de N ejemplos seleccionados aleatoriamente. La combinación de las diferentes hipótesis h_t en una hipótesis final H se genera mediante la votación mayoritaria de las h_t.

Dados (x_1, y_1), ... , (x_N , y_n), con $x_n \epsilon$ X , $y_n \epsilon$ Y = {−1, +1}, n = 1, ... , N
Para t =1,...,T :

- Seleccionar N ejemplos.
- Entrenar un aprendiz débil usando los ejemplos seleccionados.
- Obtener la hipótesis débil h_t: X → {−1, +1}.

Devolver como hipótesis final:

$$H(x) = sign\left(\frac{1}{T}\sum_{t=1}^{T} h_t(x)\right)$$

Figura 3.9: Esquema del algoritmo de *Bagging.*

Boosting

Al aplicar un algoritmo de boosting se asigna en cada iteración t un peso $D_t(x_n)$ a los ejemplos de entrenamiento. Las hipótesis h_t se crean minimizando la suma de los pesos de los ejemplos mal clasificados, así se consigue que el modelo que se aprenda en la siguiente iteración le dé más relevancia a los ejemplos mal clasificados anteriormente.
El error ϵ_t se usa para determinar los pesos $D_{t+1}(x_n)$ y la relevancia de h_t en la hipotesis final H.

Dados (x_1, y_1), ... , (x_N, y_n), con $x_n \epsilon$ X , $y_n \epsilon$ Y = $\{-1, +1\}$, n = 1, ... , N
Para t =1,...,T :

- Determinar la distribución D_t sobre los ejemplos.
- Entrenar un aprendiz débil usando distribución D_t .
- Obtener la hipótesis débil h_t : X $\rightarrow \{-1, +1\}$.
- $\varepsilon_t := Pr_{D_t}[h_t(x_n \neq y_n)] = \frac{1}{2} - \gamma_t$.

Devolver hipótesis final H, combinación de las h_t en función de los ϵ_t

Figura 3.10: Esquema del algoritmo de *Boosting*.

AdaBoost

AdaBoost se basa en estos dos métodos, pero añade un parámetro $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ al combinar las hipótesis h_t y determinar las distribuciones D_t. Por una parte, este parámetro mide la relevancia de h_t en la hipótesis final.

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Cabe destacar que $\alpha_t > 0$ si $\varepsilon_t \leq 0{,}5$, y que es mayor cuanto menor es el error, dando más importancia a las hipótesis que cometen menos errores. Por otra parte, α_t sirve para asignar los pesos $D_{t+1}(x_n)$ de cada ejemplo mediante la expresión $\exp(-\alpha_t y_n h_t(x_n))$, aumentando el peso de los ejemplos mal clasificados para tenerlos en cuenta en siguientes iteraciones. En la Figura 3.11, podemos ver un esquema del algoritmo.

Dados (x_1, y_1), ... , (x_N , y_n), con $x_n \epsilon$ X , $y_n \epsilon$ Y = {−1, +1}, n = 1, ... , N
Para t =1,...,T :

- Entrenar un aprendiz débil usando la distribución D_t.
- Obtener la hipótesis débil h_t : X → {−1, +1}.
- $\varepsilon_t := Pr_{D_t}\left[h_t\left(x_n \neq y_n\right)\right] = \frac{1}{2} - \gamma_t$.
- **Si** $\varepsilon_t = 0$ ó $\varepsilon_t \geq 0{,}5$:

  Salir del bucle ($T = t - 1$).
- $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
- **Actualizar:**

$$D_{t+1}\left(x_n\right) = \frac{D_t\left(x_n\right) \exp\left(-\alpha_t y_n h_t\left(x_n\right)\right)}{Z_t}$$

  donde Z_t es un factor de normalización tal que D_{t+1} sea una distribución de probabilidad.

Devolver como hipótesis final:

$$H\left(x\right) = sign\left(\sum_{t=1}^{T} \alpha_t h_t\left(x\right)\right)$$

Figura 3.11: Esquema del algoritmo AdaBoost.

Este parámetro hace que el *boosting* sea adaptativo, y también nos permitirá ver qué clasificadores débiles son los más relevantes y descartar los que no sean útiles (descartar *features* poco efectivas).

En cuanto a la configuración aplicada en nuestro proyecto, estamos utilizando subconjuntos de 20.000 muestras aleatorias o píxeles (10.000 por cada clase, lumen o no lumen), con tal de tener datos de todos los casos posibles en una imagen IVUS (lumen, placa, bifurcación, stent, etc.). Las iteraciones que realizamos son 150, en el primer clasificador y 300 en el segundo. Por lo tanto, en el ejemplo de la Figura 3.11, el parámetro N = 20.000 y el parámetro T = 150 y 300 respectivamente.

3.3.3. MSSL

Multi-scale Stacked Secuential Learning, es una extensión del algoritmo *Stacked Secuential Learning.* Este tipo de métodos se basan en la idea de que existe cierta relación entre los vecinos espaciales de una muestra, por lo tanto su finalidad es extraer información contextual con el fin de utilizarla para entrenar un segundo clasificador y decidir a qué clase pertenece [1].

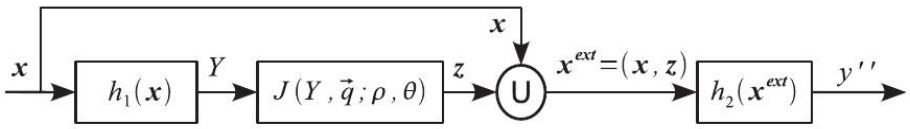


Figura 3.12: Esquema MSSL. [1]

Su funcionamiento general es simple, como podemos ver en la Figura 3.12. Un set de datos se clasifica con un primer clasificador ($h_1(x)$), obteniendo un conjunto de predicciones de pertenecer a una u otra clase para cada muestra (Y). Dicho conjunto de salida debe mantener su estructura espacial ya que sobre él se extraerá la información contextual. Existen dos maneras de extraer ésa información, una es a partir de una descomposición multi-resolución de los datos y otra es una descomposición piramidal. Nos centraremos en la descomposición multi-resolución, ya que es la que utilizamos en nuestro proyecto.
En el caso de las imágenes, se define un numero concreto de escalas a "samplear". A partir de esa variable que llamaremos s se definen s filtros gaussianos de media zero y $\sigma = \gamma^{s-1}$, donde γ es el "step" y normalmente tiene valor 2. Con esos filtros se generan s imágenes filtradas de las que se extraerán datos.
Seguidamente se lleva a cabo el mostreo o extracción de los datos. Para ello se genera un conjunto de vectores de desplazamiento que apuntarán a la posición de los datos que queremos recoger, y que conforman un patrón. Éstos vectores de desplazamiento se multiplican por un factor γ^{s-1}, lo que significa que cuanto más grande es la escala, más lejos iremos a buscar el dato a recoger. En la Figura 3.13 se puede apreciar este paso, que corresponde al segundo bloque del esquema de la Figura 3.12.

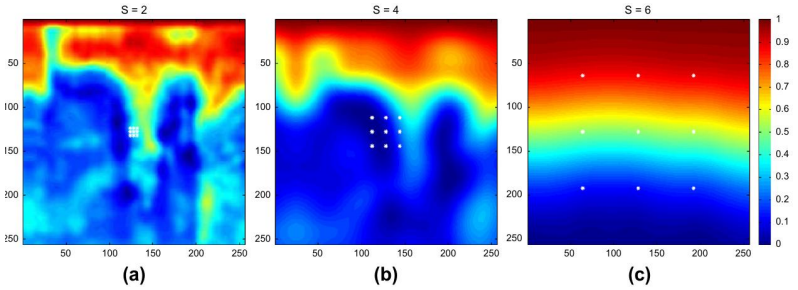


Figura 3.13: Ejemplo de descomposición multi-resolución y posible patrón de mostreo en ivus. S determina la escala. [2]

Finalmente, los nuevos datos contextuales se añaden a las características iniciales creando un conjunto extendido de features. Con estas features se testea (o se entrena) un segundo clasificador ($h_2\left(x^{ext}\right)$) para obtener finalmente un mapa de clasificación de la imagen más preciso que sólo con el primer clasificador.

MSSL 3D

Actualmente en nuestro sistema de segmentación utilizamos una versión 3D de MSSL, desarrollada por Marina Alberti [3]. Ésta versión extiende el mostreo de datos no sólo en el plano de la imagen sino sobre el eje longitudinal o temporal del pullback. Lo que significa que no sólo estaremos recogiendo información de los píxeles vecinos a uno en la misma imagen, sino en píxeles de imágenes anteriores y posteriores a la actual.
Si nos fijamos en una secuencia IVUS, podemos ver que no existen grandes variaciones entre frames sucesivos cercanos y éste hecho sirve de base para aplicar la extensión 3D de MSSL, ya que si un píxel en una imagen concreta pertenece a la clase "lumen" es muy probable que en imágenes anteriores y posteriores también pertenezca a dicha clase. En la Figura 3.14 podemos ver una descomposición 3D en 3 escalas diferentes de frames IVUS. Hay que destacar que esta extensión genera un conjunto mucho mayor de características que la versión 2D, con el coste computacional que comporta entrenar un clasificador con un número muy grande de features. Todo esto puede dificultar la implementación del algoritmo en un sistema en tiempo real, al necesitar una capacidad de cómputo mayor.

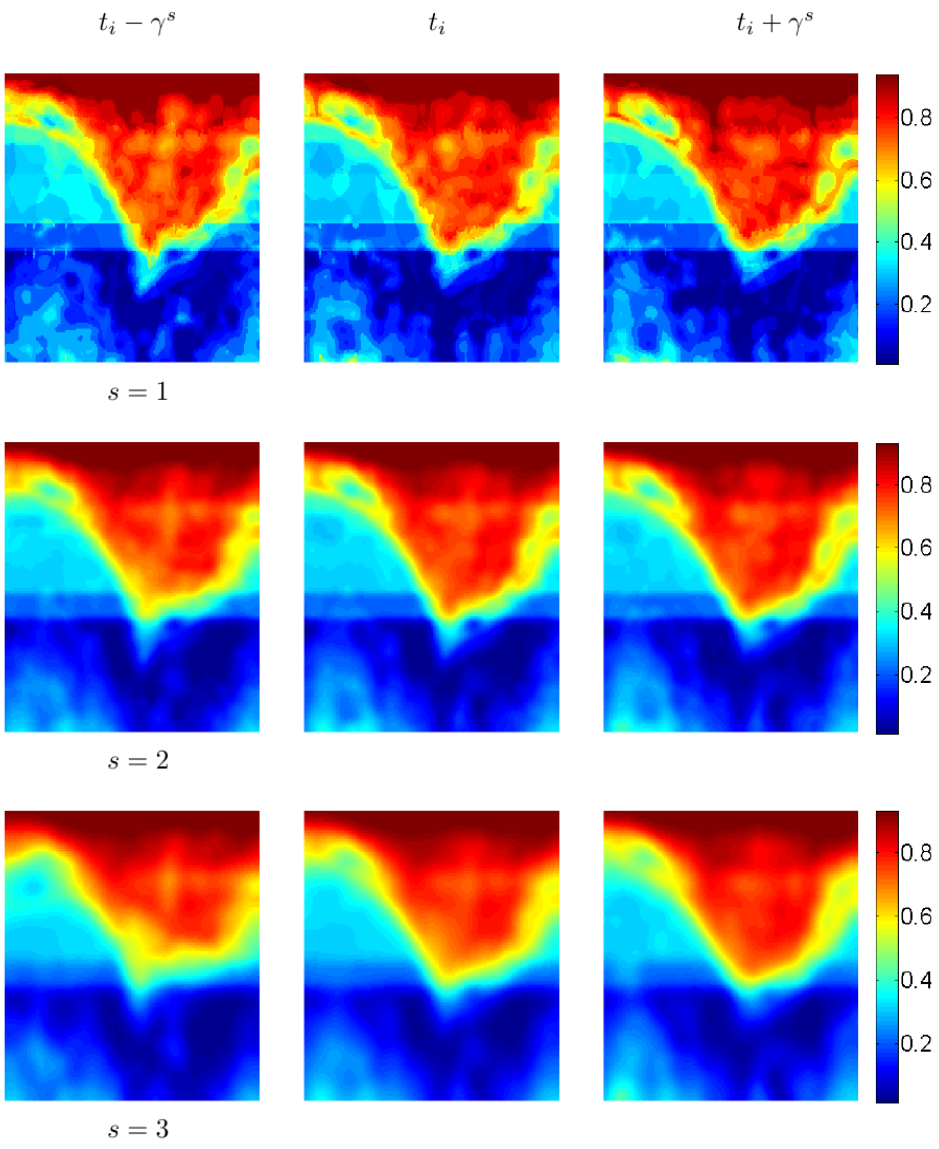


Figura 3.14: Descomposición multi-escala en tres escalas diferentes: s = 1 (primera fila), s = 2 (segunda) y s = 3 (tercera). También se muestra la descomposición de dos frames anteriores y posteriores a una distancia γ^s.

3.3.4. Active Contour Model (Snake)

Este algoritmo se aplica al mapa binario de clasificación final del algoritmo con la finalidad de determinar un borde continuo, cerrado circularmente y ajustado al mapa de clasificación. Los *Snakes* son splines que minimizan la energía,

guiados por las fuerzas de restricciones internas e influenciados por las fuerzas de restricción internas. A partir de una curva paramétrica que define el borde del lumen $C(v) = (\rho(v), \theta(v))$, que representa un *snake* con longitud de arco v, la función de energía a minimizar es la siguiente:

$$E_{acm} = \int_0^1 E_{int}(C(v)) + E_{ext}(C(v))\, dv \tag{4}$$

E_{int} es la energía interna de la *snake* debida a la flexión:

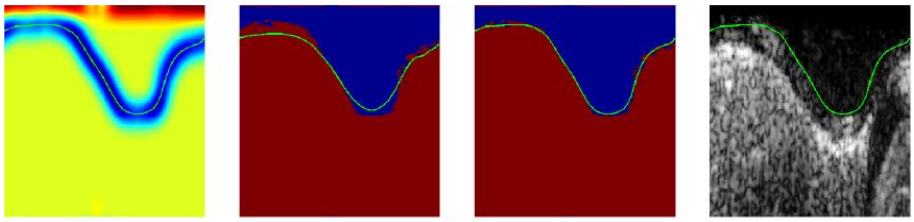
$$E_{int} = \alpha \left(\frac{\partial C(v)}{\partial v} \right)^2 + \beta \left(\frac{\partial^2 C(v)}{\partial v} \right)^2 \tag{5}$$

El primer término $\alpha \left(\frac{\partial C(v)}{\partial v} \right)^2$ representa la energía del contorno y hace que la curva se comporte como una membrana, mientras que el segundo término $\beta \left(\frac{\partial^2 C(v)}{\partial v} \right)^2$ representa la energía de curvatura y hace que se comporte como una placa delgada. Valores altos de α incrementan la energía interna y harán que se estire más, pequeños valores harán que la función de energía sea insensible a la cantidad de estiramiento. Altos valores de β harán que la energía se incremente y se generen más curvas, por el contrario, valores bajos de β harán que el *snake* sea insensible a las curvas. En nuestro proyecto éstos parámetros se han ajustado experimentalmente en 0.2 y 0.5 respectivamente.
E_{ext} representa la energía externa del *active contour model,* que se calcula con la fórmula:

$$E_{ext} = \frac{\partial (M_l(\rho,\theta) * G(0,\sigma_{acm}))}{\partial p} |(l \to nl) + \frac{\partial (M_l(\rho,\theta) * G(0,\sigma_{acm}))}{\partial \theta} |(nl \to l) \tag{6}$$

$$+ \frac{\partial (M_l(\rho,\theta) * G(0,\sigma_{acm}))}{\partial \theta} |(l \to nl)$$

donde $M_l(\rho,\theta)$ es el mapa de clasificación binario, $G(0,\sigma_{acm})$ es una gaussiana bidimensional y $(l \to nl)$ y $(nl \to l)$ representa la transición "lumen" a "no lumen" y al inverso. E_{ext} se calcula como la primera derivada del mapa de clasificación, filtrado anteriormente con una gaussiana. La definición de la curva se hace en dos pasos, un primer paso que hace una aproximación poco precisa del borde y un segundo paso que refina el resultado anterior. En la Figura 3.15 podemos ver estos pasos y su resultado final.



(a) (b) (c) (d)

Figura 3.15: Ejemplo de snake. En (a) podemos ver el mapa usado como E_{ext}; (b) y (c) son el resultado del primer y segundo paso respectivamente dibujado sobre el mapa de clasificación original; (d) es el resultado dibujado encima de la imagen original [3].

4. Mejoras aportadas al algoritmo

4.1. Introducción y configuración inicial

En esta sección expondré el trabajo que he realizado, en qué me he basado para realizar los cambios y el tipo de pruebas que he llevado a cabo para comparar los resultados posteriormente. También explicaré las medidas en las que me he basado y en qué consisten, medidas cualitativas y cuantitativas.

Inicialmente el proyecto utilizaba un conjunto de datos formado por ocho pullbacks de distintos pacientes, para hacer las pruebas más consistentes se utiliza un método LOPO (*Leave One Patient Out*), que consiste en definir el conjunto de imágenes de un paciente como conjunto de test y utilizar todos los otros como datos de training para el clasificador. Por lo tanto en cada prueba con una configuración del algoritmo determinada entrenamos ocho veces el clasificador y testeamos ocho veces sobre datos de test de un paciente diferente cada vez. De cada pullback se usan 30 frames, este valor se puede aumentar pero por problemas de falta de memoria se ha tenido que ajustar a 30.
En cuanto a la configuración inicial del clasificador, partimos de las ocho features mencionadas en la sección 3.3.1, los dos clasificadores en cascada y snake para generar el borde final del lumen. El primer clasificador realiza 150 iteraciones con un subconjunto de 10.000 muestras por cada clase (lumen y no lumen) extraídas de los 30 x 7 = 210 frames (dejamos un pullback fuera para test). El segundo clasificador, utilizando la versión 3D de MSSL, realiza 300 iteraciones con 5 escalas (5 resoluciones distintas). Por lo tanto extrae información de los 8 vecinos de cada píxel mas el pixel central, en 5 escalas y sobre los frames anteriores, actual y posteriores. Esto nos proporciona 9 x 5 x 3 = 135 características adicionales, que sumadas a las ocho iniciales nos da un total de 143 características por píxel.
Por último, los parámetros α y β del algoritmo snake tienen por valor 0.2 y 0.5 respectivamente.

4.2. Medidas del Rendimiento

4.2.1. Machine Learning

Con tal de evaluar el rendimiento de los clasificadores, he utilizado distintas medidas que explicaré a continuación.

Primero introduciremos los conceptos *True Positive*, *False Positive*, *True Negative* y *False Negative*. En el campo del aprendizaje automático (*machine learning)* estos conceptos relacionan el resultado de la clasificación con los datos clasificados manualmente (*ground-truth*), que suponemos correctos. Por ejemplo, si una muestra es positiva en el *ground-truth* y la predicción del clasificador es negativa, es un falso negativo (FN). En la Figura 4.1 podemos ver una tabla que muestra el significado de cada término.

		Predicción	
		Positivo	**Negativo**
Ground-truth	**Positivo**	True Positive (TP)	False Negative (FN)
	Negativo	False Positive (FP)	True Negative (TN)

Figura 4.1: Tabla con la relación TP, FN, FP y TN.

A partir de estos datos se obtienen distintas mediciones útiles para evaluar el rendimiento del clasificador:

- **Accuracy**: Define el porcentaje de predicciones que son correctas.

$$\frac{(TP + TN)}{(TP + TN + FP + FN)}$$

- **Sensitivity**: El porcentaje de predicciones positivas que han sido predichas como positivas.

$$\frac{TP}{(TP + FN)}$$

- **Specificity**: El porcentaje de predicciones negativas que han sido predichas como negativas.

$$\frac{TN}{(TN + FP)}$$

- **Precision**: El porcentaje de predicciones positivas correctas.

$$\frac{TP}{(TP + FP)}$$

- **False Alarm Ratio**: El porcentaje de falsos positivos.

$$\frac{FP}{(TP + FN)}$$

4.2.2. Borde

Para evaluar el borde final utilizaremos distintas medidas de error basadas en las distancias entre los bordes y la diferencia entre las áreas del borde manual y el borde segmentado automáticamente. Son medidas comúnmente utilizadas en métodos automáticos de segmentación y en el challenge IVUS internacional[6].

Estas son las distintas variables que utilizaremos para evaluar el desempeño de la segmentación:

- **Jaccard Index o Coeficiente de Semejanza de Jaccard (JACC):** es una medida estadística utilizada para comparar la semejanza o diversidad de dos conjuntos de datos. Es una medida útil para calcular la superposición de dos conjuntos de datos A y B, se expresa como un porcentaje. En nuestro caso se calcula con la siguiente ecuación, donde R_{man} y R_{auto} son las regiones de segmentación manual y automática respectivamente:

$$JI\left(R_{auto}, R_{man}\right) = \frac{|R_{auto} \cap R_{man}|}{|R_{auto} \cup R_{man}|} \tag{7}$$

- **Porcentaje de la Diferencia de Área (PAD):** es la diferencia entre las áreas de lumen automáticas (A_{auto}) y manual (A_{man}), expresado como una medida relativa a la región manual.

$$PAD = \frac{|A_{auto} - A_{man}|}{A_{man}} \tag{8}$$

- **Haussdorff Distance (HD) o Distancia de Hausdorff:** mide que lejos están dos conjuntos de datos en un espacio métrico. En nuestro caso medimos la distancia entre las dos curvas a partir de la siguiente función:

$$HD\left(C_{auto}, C_{man}\right) = \max_{\alpha \epsilon C_{auto}} \left\{ \max_{b \epsilon C_{man}} d\left(a, b\right) \right\} \tag{9}$$

  donde a y b son puntos de las curvas C_{auto} y C_{man}, respectivamente y $d\left(a, b\right)$ la distancia euclídea. El resultado es expresado en milímetros.

- **Mean Radial Distance (MRD):** es el valor medio de la distancia absoluta entre las dos curvas, calculado sobre todo el radio en la imagen polar. Se expresa en milímetros.

- **Area Error (AE):** es la diferencia entre la área de la zona de lumen calculada manualmente y automáticamente. Se expresa en milímetros cuadrados.

4.2.3. Medidas cualitativas

Para tener una visión clara del resultado de la segmentación he escogido una serie de frames concretos que muestren un resultado visualmente claro.
Para ello he buscado imágenes en los que queden claros casos de *false positive*, *false negative* y *true positive*. No he tenido en cuenta el caso de los *true negative*, ya que al ser un clasificador binario, visualmente no hay una diferencia clara entre éstos y los casos *true positive*. Dentro de estos tres casos, he buscado frames que presenten dificultades como la presencia de stent, la presencia de reflejos producidos por la guía del catéter, presencia de bifurcación y presencia de arterias cercanas. Todo esto junto con imágenes que no presentan estas dificultades. En la Figura 4.2 podemos ver ejemplos del conjunto de imágenes escogidas para observar cambios relevantes cualitativamente en las distintas configuraciones del

clasificador.

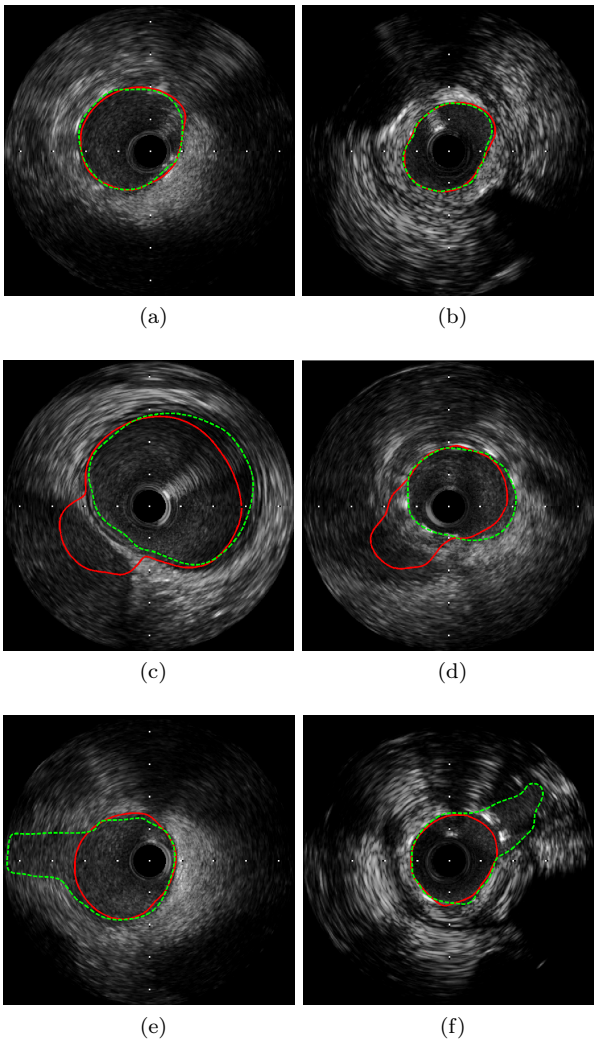


Figura 4.2: Algunos de los frames que representan los casos (a) TP; (b) TP con reflejo de la guía del catéter; (c) FP por presencia de arteria cercana; (d) FP con stent; (e) FN por bifurcación y (f) FN por bifurcación y stent.

4.2.4. Validación estadística de los datos

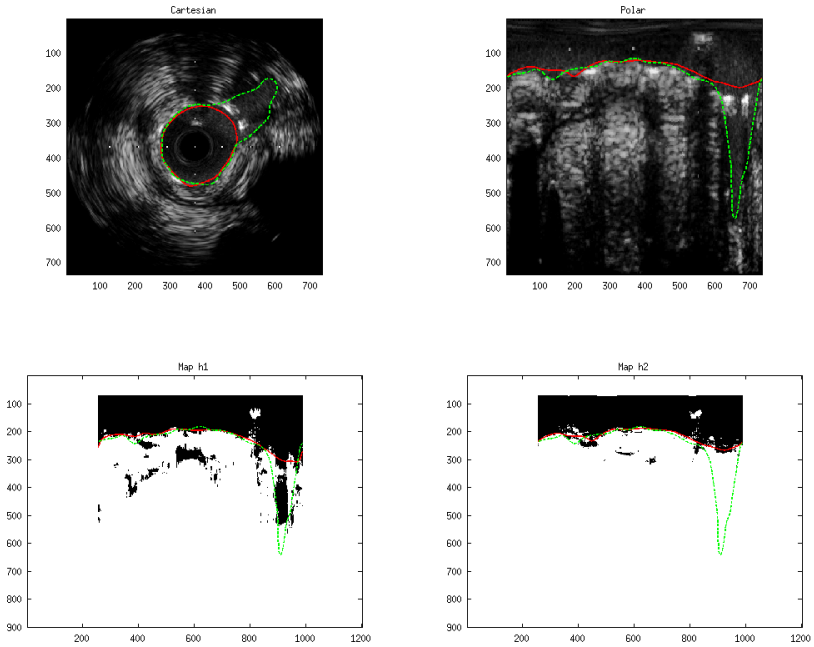
Dado que trabajamos con una cantidad grande de valores, y que son muy difíciles de comparar a simple vista, debemos validar las comparaciones entre datos estadísticamente para cerciorarnos de que realmente los valores son dis-

tintos.
Para llevar a cabo esta tarea he utilizado la "prueba de rangos con signo de Wilcoxon", es una prueba no paramétrica que sirve para comparar dos muestras y determinar si realmente son estadísticamente distintas y no hay una aparente diferencia debida al azar. No entraremos en detalle en su funcionamiento, ya que no tiene importancia en el tema que estamos tratando.
Esta prueba se lleva a cabo entre dos conjuntos de datos del mismo tipo de dos configuraciones distintas del clasificador. Por ejemplo, si se quiere determinar si el valor del False Alarm Ratio es distinto para la configuración MSSL 3D con 5 escalas y con 3 escalas, obtenemos todos los valores FAR de cada frame clasificado de la configuración 5 escalas por un lado y todos los de la configuración 3 escalas por otro. A continuación aplicamos la prueba Wilcoxon, y si el resultado es menor que 0.1 determinaremos que los dos conjuntos de datos son distintos, y por lo tanto que existe alguna diferencia real entre ellos y todo lo contrario si es mayor. Este valor significa que como máximo asumimos que el 10 % de los casos no tengan una diferencia estadísticamente significativa.
En Matlab la función que realiza este test es "ranksum", y como parámetros toma los N conjuntos de datos (en nuestro caso 2) y la variable α que hemos fijado en 0,1.

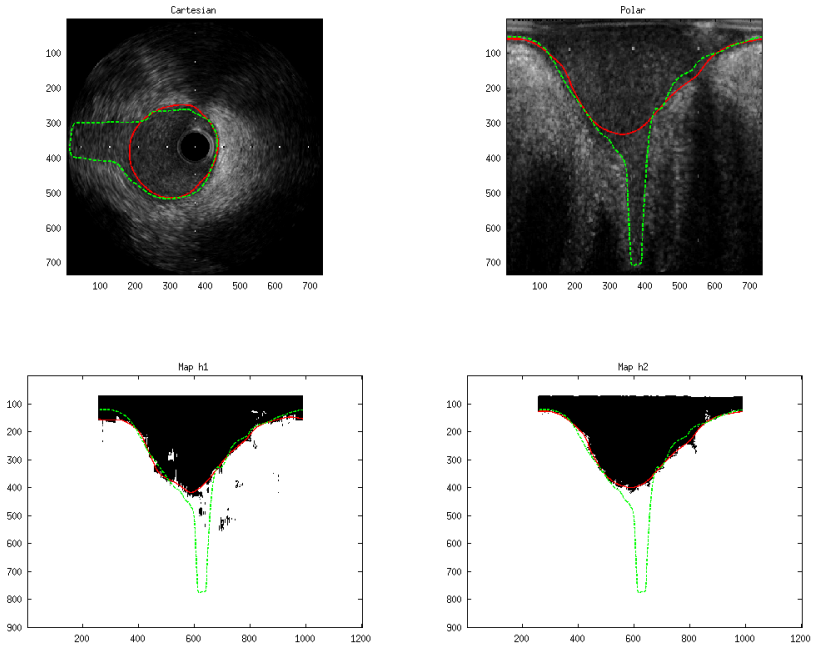
4.3. Mejoras aportadas al algoritmo

Existen múltiples aspectos a mejorar sobre el actual estado del método de segmentación de lumen. A continuación se razonarán los cambios iniciales realizados en el sistema, basándose en imágenes y datos de la configuración inicial de éste.

Como hemos visto en la Figura 4.2 (e) y (f), existe un problema claro en la detección de las bifurcaciones, ya sean imágenes con stent o sin él. Una vez comprendido el funcionamiento interno del algoritmo de segmentación, hay que fijarse en los puntos críticos donde se puede originar este fallo de segmentación. Para ello he extraído los mapas de clasificación del primer y segundo clasificador en frames en los que se da un error al detectar la bifurcación (error que es generalizado). Es un claro ejemplo de falso negativo. En la Figura 4.3 podemos ver estos mapas de dos frames con bifurcaciones.



(a) Bifurcación con stent.

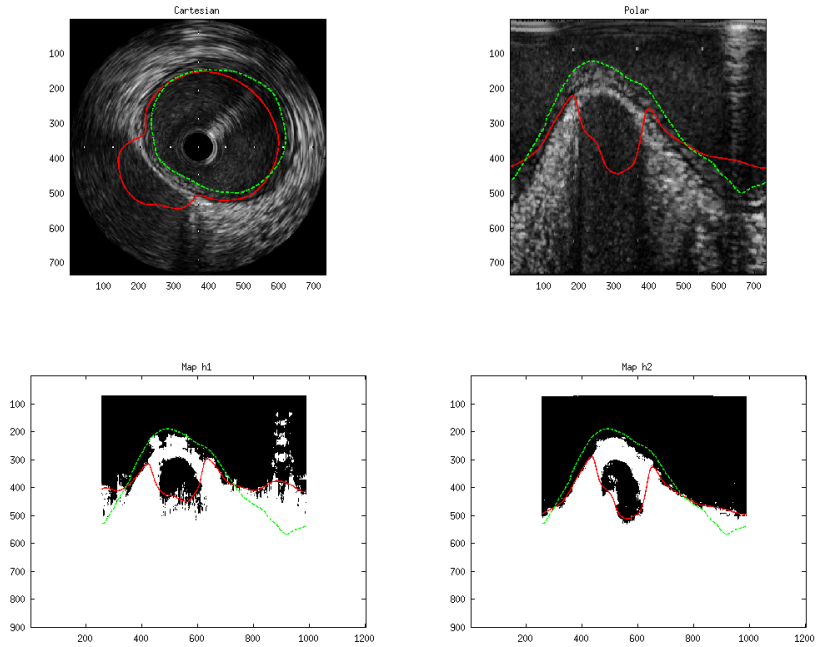


(b) Bifurcación sin stent.

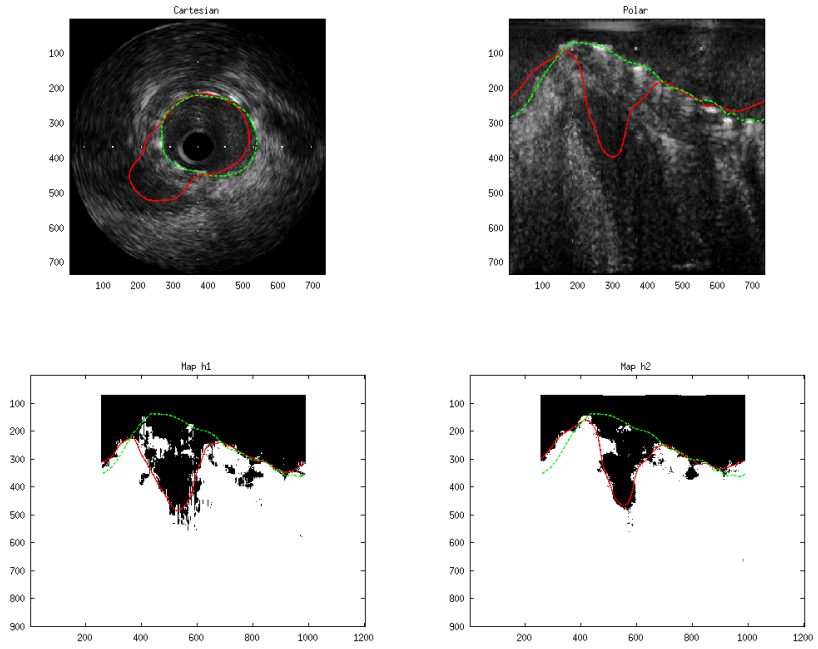
Figura 4.3: Dos ejemplos de error en la detección de una bifurcación; h1 y h2 son los mapas de clasificación del primer y segundo clasificador respectivamente. El contorno manual es de color verde, el automático de color rojo.

Como podemos ver en la imagen con stent (Figura 4.3a), el primer clasificador detecta la bifurcación pero cuando aplicamos la segunda fase de la clasificación esa zona marcada como lumen desaparece. Por lo tanto parece que un posible problema puede ser el segundo clasificador y su método MSSL 3D. Si nos fijamos en la segunda imagen (Figura 4.3b), vemos que la bifurcación no se detecta ni en el primer clasificador, lo que nos sugiere que puede ser un problema de features, puede que no sean suficientemente buenas para extraer información relevante de la zona de bifurcación. Otra posible explicación es que puede no haber suficientes muestras de frames con bifurcación en el conjunto de test, ya que el caso con stent parece detectarse mejor que el caso sin stent, por lo menos en la primera clasificación.
De este primer caso hemos extraído como objetivos de estudio el algoritmo **MSSL 3D**, el conjunto de **features iniciales** y la posible falta de **muestras para los casos de bifurcaciones**.

Un segundo error importante es la detección incorrecta de zonas de lumen, o lo que es lo mismo, zonas de falsos positivos. Para ilustrar esta dificultad, mostraremos, como en el caso anterior, los mapas de clasificación de dos casos en los que se detecten zonas de lumen donde no las hay (Figura 4.4).
Fijémonos en la primera imagen (Figura 4.4a), es un caso claro de falso positivo en un frame sin stent. La primera clasificación muestra un "agujero" marcado como lumen en una zona en la que hay una arteria cercana, la segunda clasificación consigue un resultado ligeramente mejor pero el *snake* es demasiado flexible y envuelve con el borde la zona. En la segunda imágen (Figura 4.4b), que es de una arteria con stent, tenemos un error similar. En este caso es problema de los dos clasificadores, el primero clasifica una zona que claramente no es lumen como lumen, y el segundo no consigue un resultado mejor. El problema principal es el primer clasificador, ya que el segundo se basa en el primero, por lo tanto tenemos otra razón para estudiar las **features**, ya que en el conjunto de training tenemos muchos casos de este tipo y además es un error que se observa en bastantes frames de test. También podrían ajustarse nuevamente los parámetros del **snake**, para reducir su flexibilidad, aunque es un punto muy sensible ya que esto podría afectar a la mayoría de frames bien clasificados. El valor cuantitativo más relacionado con este caso es el False Alarm Ratio, por lo tanto debemos intentar reducir especialmente ese valor.



(a) FP sin stent.



(b) FP con stent.

Figura 4.4: Dos ejemplos de mapas de clasificación falsos positivos; h1 y h2 son los mapas de clasificación del primer y segundo clasificador respectivamente. En verde la segmentación manual, en rojo la automática.

En resumen, tenemos varios puntos sobre los que intentar mejorar, que son: el algoritmo MSSL 3D, el conjunto de features, la falta de datos de training de bifurcaciones y posiblemente un nuevo ajuste del *snake*. Para reducir estos errores tendremos en cuenta todos los valores cuantitativos, tanto valores de aprendizaje automático (accuracy, precision, false alarm ratio, etc.) como valores de error del borde (distancia de Hausdorff, índice Jaccard, etc.), con la intención de mejorarlos o por lo menos mantenerlos.

4.4. Cambios y pruebas

En esta sección se describen los cambios y las pruebas que se han realizado al sistema. Se subdivide en tres subsecciones, cada una centrándose en uno de los objetos de estudio mencionados anteriormente.

4.4.1. Features

Las características iniciales son la base para el entrenamiento del primer clasificador, los ocho filtros de extracción de características iniciales parecen dar un buen resultado general, pero como hemos visto en la sección 4.3 existen ciertos casos en los que su rendimiento es mejorable. Con tal de intentar mejorar en este aspecto, he añadido dos filtros nuevos al conjunto inicial. El primer filtro ha sido utilizado en otros proyectos de segmentación IVUS [8], se trata de un filtro que he denominado "Rayleigh", ya que se basa en la distribución de Rayleigh . El segundo filtro es una diferencia del nivel de gris entre la zona inferior y superior del píxel ponderada, este filtro ha sido ideado por mi.

Los dos filtros han sido implementados en Matlab, y con tal de no incrementar en exceso el tiempo de entrenamiento y de test del sistema, y agilizar las pruebas los he convertido a lenguaje C. En el anexo (sección 8.2), incluyo un pequeño manual para convertir automáticamente código Matlab a C y compilar archivos "mex" utilizando una herramienta que incluye la versión 2012 de Matlab.

Filtro Rayleigh

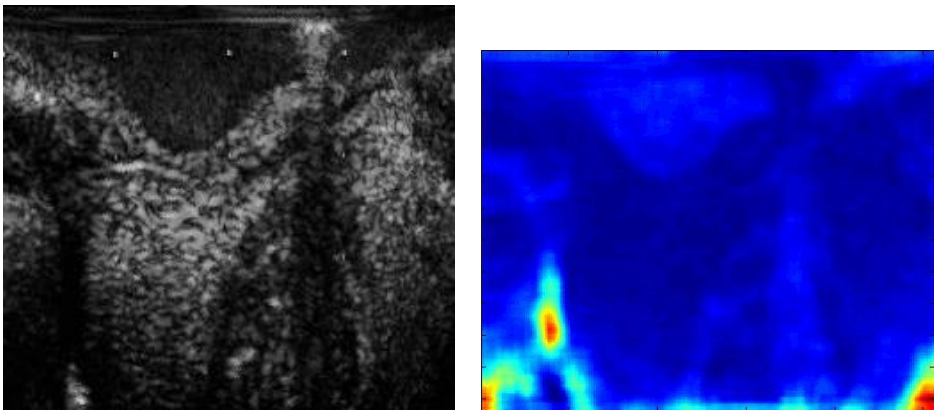
La distribución estadística local de una imagen ultrasonora es de tipo Rayleigh. Se ha demostrado que diferentes tejidos tienen parametros de R. distintos. Bajo esta hipotesis queremos diferenciar la sangre de los tejidos que lo rodean analizando los parametros de R, osea su propiedades estadísticas [4, 5].
La función de densidad de probabilidad de Rayleigh, que depende del parámetro σ es la siguiente:

$$f\left(x|\sigma\right) = \frac{x \exp\left(\frac{-x^2}{2\sigma^2}\right)}{\sigma^2} \tag{10}$$

Y dado un conjunto de datos x la estimación de máxima verosimilitud del parámetro se calcula con:

$$\sigma \approx \sqrt{\frac{1}{2N}\sum_{i=0}^{N} x_i^2} \tag{11}$$

El filtro implementa una ventana deslizante de tamaño X sobre toda la imagen, y por cada bloque calcula el histograma de los píxeles que se encuentran en él. A partir de ese histograma se aplica la función 11 con tal de obtener el parámetro σ, que es el valor que le daremos al píxel central de la ventana y que será la la feature asociada a ese píxel. En la Figura 4.5 podemos ver un ejemplo de salida del filtro.



(a) Imagen original. (b) Imagen filtrada.

Figura 4.5: Ejemplo de salida del filtro Rayleigh.

Filtro de diferencia de nivel de gris ponderada

Este filtro trabaja sobre la imagen polar verticalmente. Por cada píxel realiza la diferencia del valor medio de nivel de gris de los píxeles inferiores a él y los píxeles superiores, y este valor se multiplica por la media del nivel de gris de la columna anterior, ajustando así este valor dependiendo de la columna anterior. En la Figura 4.6 podemos ver un esquema de cómo funciona este filtro.

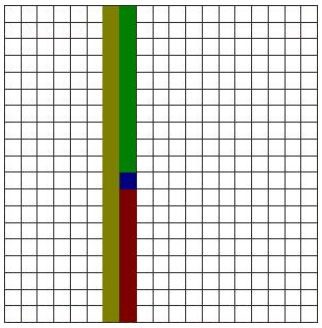
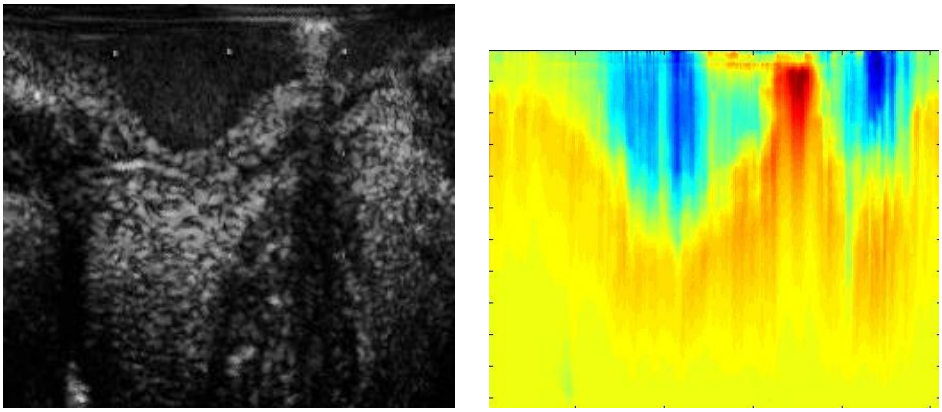


Figura 4.6: Esquema del filtro de diferencia de nivel de gris ponderada. En rojo los píxeles inferiores y en verde los superiores, en amarillo la columna anterior.

Es un filtro basado en los filtros Shade y relative Shade(sección 3.3.1). La idea principal es que en imágenes polares la zona de lumen siempre está arriba y tiene un color bastante uniforme y más oscuro que la zona de no lumen, por lo tanto en el borde que separa estas dos zonas encontraremos un valor menor de esta diferencia. La multiplicación por la media de la columna anterior, se realiza con la finalidad de ponderar el valor para tener en cuenta posibles cambios bruscos en el borde o artifacts tales como reflejos. Así, si hay un reflejo en la columna anterior se verá reflejado en los valores de los píxeles de la siguiente columna. En la Figura 4.7 podemos ver un ejemplo de salida del filtro.



(a) Imagen original (b) Imagen filtrada.

Figura 4.7: Ejemplo de imagen filtrada con diferencia de nivel de gris ponderada.

Pruebas

Se han realizado pruebas con distintas configuraciones de features, manteniendo el resto de las configuraciones del método tal y como era originalmente.
En concreto he definido cuatro configuraciones distintas:

1. Ocho features iniciales mas filtro de diferencia de nivel de gris ponderada, en total 9 features.
2. Ocho features iniciales mas filtro de Rayleigh, con un tamaño de ventana deslizante de 14 x 14 px, en total nueve features.
3. Ocho features iniciales, filtro de diferencias y tres filtros Rayleigh con tamaños distintos de ventana (6 x 6, 10 x 10 y 14 x 14 píxeles). En total doce features.
4. Ocho mejores features.

Las dos primeras configuraciones me han servido para comprobar la validez de estas nuevas features separadamente, al poder compararlas con la configuración original. La tercera configuración es útil para determinar que características son las más relevantes, a parte de comparar su rendimiento con la configuración original. Esto es posible gracias a que el clasificador se genera con AdaBoost, y utilizando los parámetros α del clasificador final podemos generar un histograma con las features que más se utilizan, pudiendo descartar así las menos relevantes.
Finalmente a partir del histograma de características más relevantes, y con el fin de no aumentar la complejidad y el tiempo de ejecución, he seleccionado las ocho mejores (cuarta configuración) y he realizado otro test para poder determinar que esta configuración final sigue mejorando o igualando las anteriores.

En la sección 5.1 se muestran los resultados de estas pruebas.

4.4.2. MSSL

La versión 3D de MSSL parece obtener buenos resultados, pero si la analizamos en profundidad hay algunos puntos un tanto incoherentes en este método.

Actualmente la implementación de este algoritmo extrae valores de probabilidad de frames anteriores y posteriores al actual, pero cabe hacerse una pregunta importante. ¿A qué distancia del frame estamos extrayendo información?. Si miramos el código vemos que el vector de desplazamiento se calcula de la siguiente manera: $delta = 2^{scale-1}$, esto significa que a cada escala estamos sampleando datos de píxeles a una distancia $2^{scale-1}$ del píxel en el plano (mismo frame), pero estamos observando píxeles a la misma distancia temporal. Es decir, si estamos extrayendo los datos de la quinta escala, estamos mirando píxeles a 2^4 frames hacia adelante y 2^4 hacia atrás. Obviamente, un dato de un frame a una distancia de 16 frames tiene poca relación con el frame actual.

Para solventar esto he cambiado en el código la distancia longitudinal a la que se samplean los datos, creando una nueva "configuración" en la que el vector de desplazamiento en el eje temporal ahora es $delta = scale$. Así, por ejemplo, en una escala 3 estaremos observando los dos frames anterior y posterior a una distancia 3. A esta versión la he llamado "3D Adyacente" con tal de diferenciarla de la original, y he realizado tests con esta modificación sin modificar el resto de la configuración del clasificador para comprobar que no perdíamos efectividad. Por falta de tiempo no he podido implementar otro cambio en el algoritmo para que, en vez de utilizar los frames gated adyacentes, se utilicen frames adyacentes del pullback original, ya que son mas próximos y pueden tener más relación entre ellos. En la sección 7 se detallará este posible cambio para investigaciones futuras.

Otro aspecto a tener en cuenta de esta versión 3D es que incrementa considerablemente la complejidad del método por varias razones, la primera es que al depender de imágenes posteriores a la actual se hace imposible cualquier futura implementación en un sistema de segmentación a tiempo real. La segunda es que incrementa considerablemente el tiempo de ejecución del proceso de training y test del clasificador, ya que genera un número alto de características que ralentizan el clasificador (135 por cada píxel de la imagen para 5 escalas).
Estos motivos me han llevado a probar una versión 2D de MSSL, para ver si realmente en 3D se obtiene una mejora considerable hasta el punto en que valga la pena ese coste computacional.

Finalmente, un parámetro que he querido testear es el número de escalas, concretamente he hecho pruebas con 5 escalas y 3 escalas para cerciorarme de que son necesarias las 5 escalas actuales.

Resumiendo estas son las configuraciones que he testeado:

1. 3D Original con 5 Escalas.
2. 3D Original con 3 Escalas.
3. 3D Adyacente con 5 Escalas.
4. 3D Adyacente con 3 Escalas.
5. 2D con 5 Escalas.
6. 2D con 3 Escalas.

En la sección 5.2 se muestran los resultados relevantes de las pruebas con estas modificaciones y una tabla con los tiempos de training y test de la versión 2D y 3D.

4.4.3. Bifurcaciones

Como hemos observado en la sección 4.3, las bifurcaciones a veces no están detectadas por el primer clasificador.
Si miramos el conjunto de training que hemos usado, vemos que no hay muchos frames con bifurcaciones para que el algoritmo pueda aprender de ellos, en concreto tenemos 11 frames con bifurcaciones de un total de 240 (4,5 %). Además al hacer un subsampling de las muestras para entrenar el clasificador (20.000 muestras aleatorias), es difícil probabilísticamente que se escojan suficientes muestras para poder tener en cuenta este caso especial.

Una posible solución es añadir muestras con bifurcaciones al conjunto de training. Para ello he extraído imágenes con bifurcaciones de tres pullbacks de dos pacientes distintos, obteniendo 83 frames independientes. Al no ser frames con una relación temporal, no son adyacentes, sólo he podido hacer pruebas con la configuración 2D de MSSL que no tiene en cuenta los frames anteriores y posteriores.
Las pruebas realizadas han sido con cuatro configuraciones distintas:

1. Sólo las bifurcaciones como conjunto de training, MSSL 2D a 5 escalas y ocho mejores features.

2. Los ocho pullbacks iniciales mas el nuevo con bifurcaciones, MSSL 2D a 5 escalas y ocho mejores features.

En cuanto a las features, he utilizado las ocho mejores ya que en los resultados del apartado 3.3.1, veremos que se obtiene el mejor resultado de todas las configuraciones posibles.

Otro posible aspecto a tener en cuenta son los parámetros del algoritmo que genera el *snake,* pero por falta de tiempo no he podido profundizar en el tema. En la sección 7 se menciona como posible objeto de estudio para solventar la detección de bifurcaciones.

En la sección 5.3, se exponen los resultados de estas pruebas y sus comparaciones.

5. Resultados

En este apartado se muestran los resultados de las pruebas realizadas. Primero mostraremos en tablas los valores referentes al clasificador y al borde (datos cuantitativos), después, si es necesario, se mostrarán imágenes de frames críticos para ver si cualitativamente obtenemos alguna mejora. Finalmente se comentarán los resultados.
Todas las comparaciones han sido comprobadas estadísticamente con el test Wilcoxon con una $\alpha = 0{,}1$, los valores que son estadísticamente distintos los marcaremos con un asterisco.

5.1. Análisis de Features

5.1.1. Diferencia de Nivel de Gris Ponderada

Vamos a comparar la configuración original del sistema de segmentación (3D MSSL, 5 escalas) contra la misma configuración añadiendo la feature de diferencia de nivel de gris ponderada.

En la tabla de la Figura 5.1a se muestran los valores de rendimiento de los dos clasificadores (H1 y H2).

	8 Features	9 Features
%	mean±std	mean±std
H1		
Accuracy	93,47±3,03	93,75±2,48
Sensitivity	92,60±5,64	92,42±5,90
Precision	94,04±3,70	94,40±2,85
Specificity	82,92±11,57	83,55±10,57
FAR	21,71±19,10	20,22±16,46
H2		
Accuracy*	95,49±2,40	96,04±2,06
Sensitivity	91,48±6,48	91,44±6,93
Precision*	96,45±3,13	97,39±1,95
Specificity*	89,88±8,80	91,98±6,81
FAR*	11,13±10,74	8,42±7,98

(a)

	8 Features	9 Features
	mean±std	mean±std
JACC(%)*	81,29±10,96	83,24±9,99
HD(mm)	0,60±0,55	0,50±0,47
AE(mm^2)	1,10±1,13	0,92±1,01
PAD(%)*	10,95±8,915	9,77±8,932
MRD(mm)*	0,17±0,11	0,14±0,08

(b)

Figura 5.1: Comparativa diferencia de gris ponderada; (a) valores *de machine learning; (b) valores del borde.*

En el primer clasificador parece haber una mejora en ***specificity*** y en en ***false alarm ratio***, sin embargo estadísticamente esta comparación no es relevante. Los demás valores parecen no variar respecto a la configuración original. En el segundo clasificador obtenemos mejores resultados, aumenta el porcentaje de ***accuracy***, ***precision*** y **specificity** considerablemente. Además el valor

FAR desciende en casi tres puntos (menor es mejor).
Veamos ahora los valores referentes al borde en la tabla de la Figura 5.1b. Para entender mejor la comparativa hay que recordar que los valores del índice Jaccard son mejores cuanto mas altos, los demás son mejores cuanto más bajos. Los valores de *Hausdorff distance* y *area error* no nos dan información relevante. En cuanto al índice Jaccard, PAD y MRD obtenemos mejores resultados que con la configuración original. Por lo tanto podemos decir que este filtro nos es útil, ya que estamos reduciendo la tasa de falsos positivos considerablemente y mejorando la mayoría de indicadores, nunca empeorando.

5.1.2. Filtro Rayleigh

Comparamos ahora el filtro Rayleigh con la configuración original de features.

	8 Features	9 Features
%	mean±std	mean±std
H1		
Acuracy*	93,47±3,03	94,12±2,95
Sensitivity	92,60±5,64	92,89±5,19
Precision*	94,04±3,70	94,80±3,44
Specificity*	82,92±11,57	84,85±10,89
FAR*	21,71±19,10	18,82±17,56
H2		
Accuracy*	95,49±2,40	95,03±4,29
Sensitivity*	91,48±6,48	92,85±6,32
Precision	96,45±3,13	94,90±7,89
Specificity	89,88±8,80	89,63±9,50
FAR	11,13±10,74	12,18±12,90

(a)

	8 Features	9 Features
	mean±std	mean±std
JACC(%)	81,29±10,96	82,09±11,48
HD(mm)	0,60±0,55	0,68±0,75
AE(mm^2)	1,10±1,13	1,84±2,97
PAD(%)	10,95±8,91	14,55±16,63
MRD(mm)*	0,17±0,11	0,18±0,18

(b)

Figura 5.2: Comparativa Rayleigh; (a)valores *de machine learning*; (b) valores del borde.

En este caso vemos claramente que en el primer clasificador obtenemos mejoras en casi todos los valores excepto el valor ***sensitivity***. En el segundo clasificador no hay diferencias tan claras, perdemos medio punto en ***accuracy*** , un punto en **FAR** y ganamos un punto en ***sensitivity***.
Veamos los valores del borde.
En los valores del borde, el único valor relevante en este caso es el ***mean radial distance***, que empeora muy ligeramente. Este filtro no parece aportar una característica relevante porque aunque mejora en la primera clasificación, la segunda no mejora y los resultados del borde empeoran ligeramente, pero esto último puede ser debido a la curva *snake*. Al ser un filtro que depende del

tamaño de la ventana deslizante que le demos, le daremos otra oportunidad y en la siguiente sección probaremos tres tamaños distintos de ventana. De esta manera podremos ver si el clasificador realmente utiliza alguno de estos tres filtros, sino lo descartaremos.

5.1.3. Ocho Mejores Features

En este apartado mostraremos resultados de dos configuraciones de features distintas. Primero veremos los resultados de la configuración con 12 features (ocho iniciales, diferencia de gris y tres filtros Rayleigh con tres tamaños de ventana distintos), despúes escogeremos las ocho mejores para comprobar si dan mejores resultados que las ocho iniciales.

12 Features

	8 Features	12 Features
%	mean±std	mean±std
H1		
Acuracy*	93,47±3,03	94,10±2,64
Sensitivity	92,60±5,64	92,76±5,47
Precision*	94,04±3,70	94,86±2,87
Specificity	82,92±11,57	84,71±10,2
FAR	21,71±19,10	18,64±15,82
H2		
Accuracy*	95,49±2,40	96,49±1,84
Sensitivity	91,48±6,48	92,14±6,49
Precision*	96,45±3,13	97,67±2,13
Specificity*	89,88±8,80	93,18±6,04
FAR*	11,13±10,74	7,11±6,81

(a)

	8 Features	12 Features
	mean±std	mean±std
JACC(%)*	81,29±10,96	85,08±9,47
HD(mm)*	0,60±0,55	0,46±0,47
AE(mm^2)*	1,10±1,13	0,90±1,01
PAD(%)*	10,95±8,91	9,25±8,46
MRD(mm)*	0,17±0,11	0,13±0,08

(b)

Figura 5.3: Comparativa 12 features; (a) valores *de machine learning*; (b) valores del borde.

Aquí tenemos claro que la combinación de estas doce características mejora muy notoriamente los valores del segundo clasificador, en especial la ***specificity*** (1,8 %) y el ***false alarm ratio*** (-4,02 %). Los datos del primer clasificador también obtienen mejores resultados. Si comprobamos los datos del borde (Figura 5.3b), se corrobora esta mejora, todos los valores han pasado el test de Wilcoxon y todos han mejorado significativamente, lo que significa que en la segmentación final obtenemos resultados más afinados. Tomaremos esta configuración como la mejor hasta el momento.

Ocho mejores features
Con tal de no incrementar el número inicial de features hemos escogido, en la

configuración anterior de 12 features, las ocho que más relevancia tienen en el primer clasificador.
Para ello hemos observado los clasificadores resultantes de las ocho rondas de training y hemos extraído las ocho con un parámetro α mayor, a partir de esa información hemos creado el histograma de la Figura 5.4 sumándolas todas. Los valores del eje "x" corresponden al número de feature, los del eje "y" el número de veces que aparecen en los resultados. También hemos calculado la suma de los pesos (α) de cada feature de cada uno de los ocho clasificadores (Figura 5.5).
La numeración de las features es la siguiente:

1. Imagen original
2. Filtro de Gabor 1
3. Filtro de Gabor 2
4. Filtro de Gabor 3
5. Filtro de Gabor 4
6. Shade
7. Relative Shade
8. Correlación de Pearson.
9. Diferencia de nivel de gris ponderada.
10. Rayleigh, ventana de 6 x 6 px.
11. Rayleigh, ventana de 10 x10 px.
12. Rayleigh, ventana de 14 x 14 px.

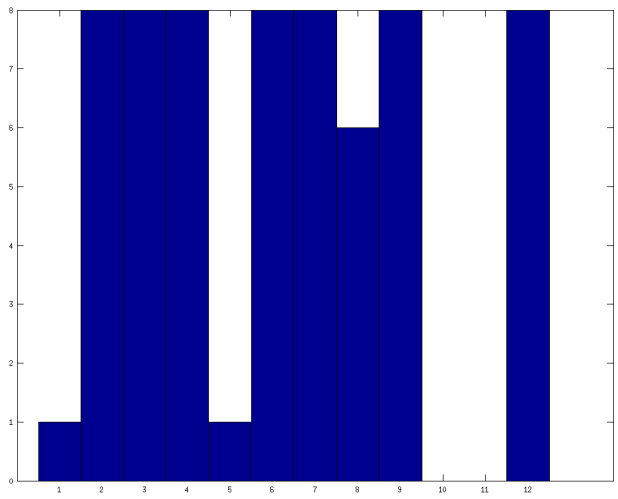


Figura 5.4: Histograma de las 8 mejores features.

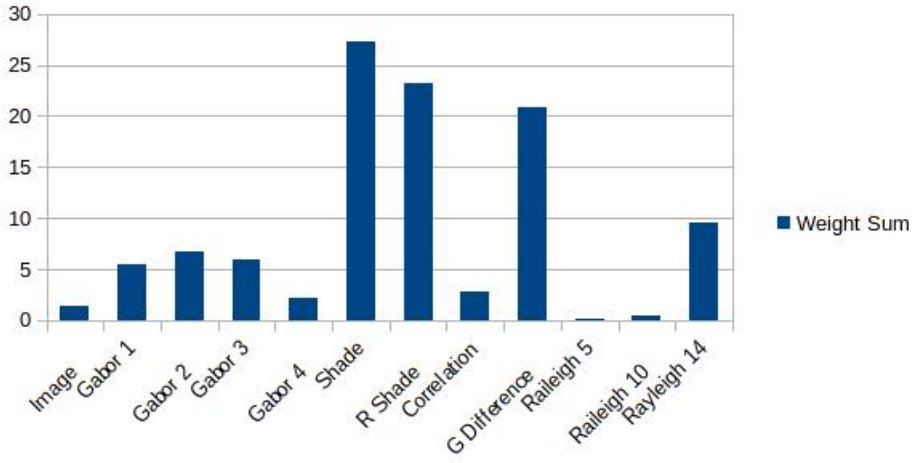


Figura 5.5: Gráfica de la suma de los pesos de las features de cada uno de los ocho clasificadores resultantes.

Como vemos en el histograma debemos descartar la imagen original, el filtro de Gabor 4 y las Rayleigh (6 x 6 y 10 x 10). Si nos fijamos en la gráfica de la suma de los pesos vemos que la **nueva feature de Rayleigh sí nos es útil** con un tamaño de ventana de 14 x 14 px, ya que tiene un peso considerable respecto a las demás. Tomamos como configuración final:

1. Filtro de Gabor 1
2. Filtro de Gabor 2
3. Filtro de Gabor 3
4. Shade
5. Relative Shade
6. Correlación de Pearson.
7. Diferencia de nivel de gris ponderada.
8. Rayleigh 14 x 14.

Una vez escogida esta configuración de features, volvemos a testear. En la Figura 5.6 vemos los resultados comparados con la versión de 12 características, ya que es la mejor configuración hasta el momento y no queremos perder rendimiento respecto a ella.

	12 Features	8 Mejores
%	mean±std	mean±std
H1		
Acuracy	94,10±2,64	94,21±2,50
Sensitivity	92,76±5,47	92,69±5,23
Precision	94,86±2,87	94,98±2,77
Specificity	84,71±10,2	85,00±10,18
FAR	18,64±15,82	18,23±15,73
H2		
Accuracy	96,49±1,84	96,54±1,85
Sensitivity	92,14±6,49	92,45±6,30
Precision	97,67±2,13	97,68±1,85
Specificity	93,18±6,04	92,91±6,34
FAR*	7,11±6,81	7,45±7,39

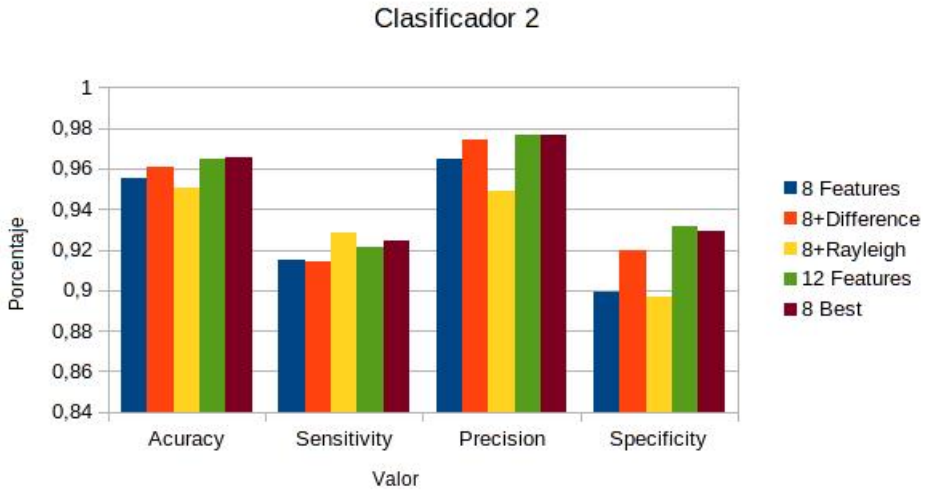
(a)

	12 Features	8 Mejores
	mean±std	mean±std
JACC(%)	85,08±9,47	85,07±9,54
HD(mm)	0,46±0,47	0,451±0,45
AE(mm^2)	0,90±1,01	0,84±0,95
PAD(%)	9,25±8,46	8,90±8,44
MRD(mm)	0,13±0,08	0,12±0,07

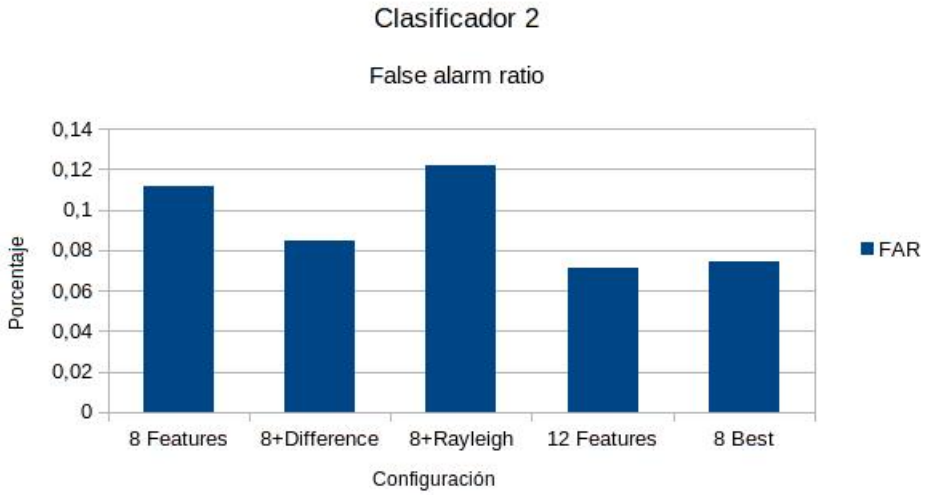
(b)

Figura 5.6: Comparativa 8 mejores features; (a) valores *de machine learning*; (b) valores del borde.

Los valores de machine learning apenas varían, y los del borde varían mejorando ligeramente. Los tests de Wilcoxon no dan resultados positivos, por lo que asumimos que no hay diferencias significativas en las dos configuraciones y **damos esta última como válida**. Podemos comparar los resultados del segundo clasificador con las cinco configuraciones en los gráficos de la Figura 5.7.



(a)

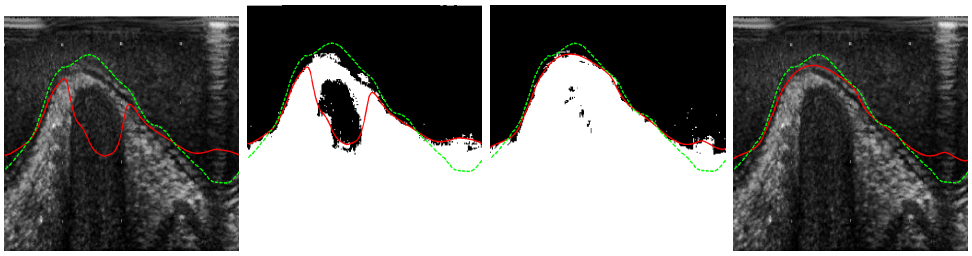


(b)

Figura 5.7: Gráficos de los valores de machine learning del clasificador 2.

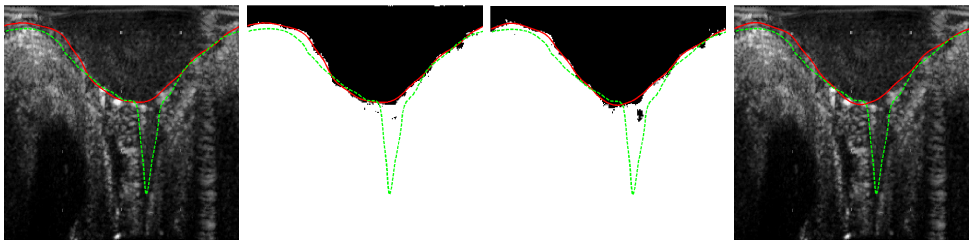
Cualitativamente podemos observar mapas de clasificación de frames críticos en casos FP, FN y TP. Como vemos, el falso positivo (Figura 5.8) ha mejorado drásticamente, clasifica bien casi toda la zona que antes era marcada como lumen sin serlo. En cuanto al caso de falso negativo (Figura 5.9), no vemos ningún cambio destacable, no conseguimos detectar la bifurcación. El caso TP (Figura

5.10) no presenta cambios mayores, excepto un mapa mas suavizado y un borde un poco más preciso.



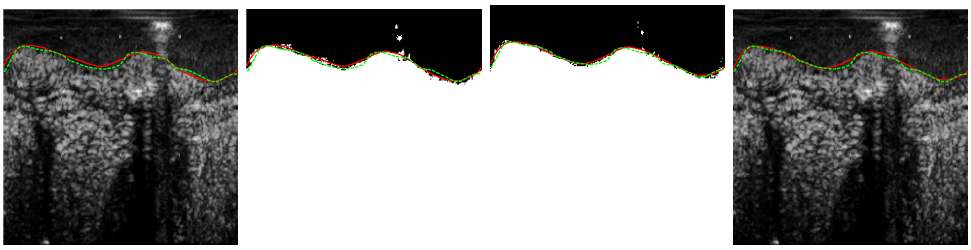
(a) (b) (c) (d)

Figura 5.8: FP features originales contra ocho mejores, los mapas de clasificación son del segundo clasificador. (a) imagen original; (b) mapa original; (c) mapa nueva configuración; (d) imagen segmentada con la nueva configuración.



(a) (b) (c) (d)

Figura 5.9: FN features originales contra ocho mejores, los mapas de clasificación son del segundo clasificador. (a) imagen original; (b) mapa original; (c) mapa nueva configuración; (d) imagen segmentada con la nueva configuración.



(a) (b) (c) (d)

Figura 5.10: TP features originales contra ocho mejores, los mapas de clasificación son del segundo clasificador. (a) imagen original; (b) mapa original; (c) mapa nueva configuración; (d) imagen segmentada con la nueva configuración.

5.2. MSSL 2D vs MSSL 3D

A continuación mostraremos los resultados de las distintas configuraciones de MSSL.

3D Adyacente
Antes que nada debemos comprobar si la versión MSSL 3D adyacente es, por lo menos, equivalente a la versión 3D original. Las pruebas se han hecho sobre la versión a 5 escalas para las dos configuraciones, y sobre las ocho features originales. Cabe recordar que los valores del primer clasificador son los mismos, ya que estamos evaluando el rendimiento del segundo estadio de la clasificación.

	3D original	3D adyacente
%	mean±std	mean±std
H2		
Accuracy	95,49±2,40	95,67±2,20
Sensitivity	91,48±6,48	90,41±7,29
Precision*	96,45±3,13	96,95±2,92
Specificity*	89,88±8,80	91,36±7,99
FAR*	11,13±10,74	9,20±9,33

Figura 5.11: Comparativa MSSL 3D original contra adyacente, datos de *machine learning* del segundo clasificador.

En la tabla de la figura vemos que en especial el **FAR** desciende casi dos puntos y el valor **specificity** aumenta alrededor de un punto y medio, además pasan el test Wilcoxon, por lo tanto son realmente valores relevantes. Los demás valores son ligeramente superiores, exceptuando el valor **sensitivity** que desciende. Con estos datos determinamos que la nueva versión adyacente da mejores resultados que la versión original, a partir de ahora **descartamos la versión original**.

3D Adyacente vs 2D

	3D	2D
%	mean±std	mean±std
H2		
Accuracy	95,67±2,20	95,89±2,26
Sensitivity	90,41±7,29	90,84±6,73
Precision	96,95±2,92	97,35±2,51
Specificity	91,36±7,99	92,05±8,04
FAR	9,20±9,33	8,56±9,84

Figura 5.12: Comparativa MSSL 3D adyacente contra 2D, datos de *machine learning* del segundo clasificador.

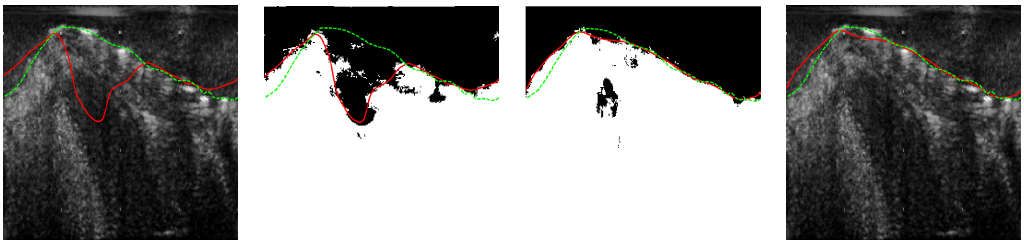
En este caso obtenemos unos resultados muy similares, además no son estadísticamente distintos por lo que determinamos que los dos métodos **son equivalentes**.

Aunque a nivel de datos de rendimiento de clasificación sean equivalentes, cabe recordar que estamos obteniendo una mejora muy considerable en cuanto a coste computacional, ya que estamos reduciendo el número de features extendidas de 143 a 53.
Veamos como afecta este hecho a los tiempos de ejecución del algoritmo en la tabla (Figura 5.13), los valores son la media de las ocho rondas de test y de training respectivamente. Cabe aclarar que el tiempo de test es el tiempo que tarda en clasificar un pullback (30 frames), el tiempo de training es el que tarda en entrenar el clasificador con los siete pullbacks restantes. Como podemos observar, es una mejora drástica, estamos reduciendo a mucho más de la mitad los tiempos de ejecución.

	3D a 5 escalas	2D a 5 escalas	Aumento de rendimiento
	mean	mean	percent
Test (segundos)	89,11	8,3666	90
Training (segundos)	719,25	249,64	65

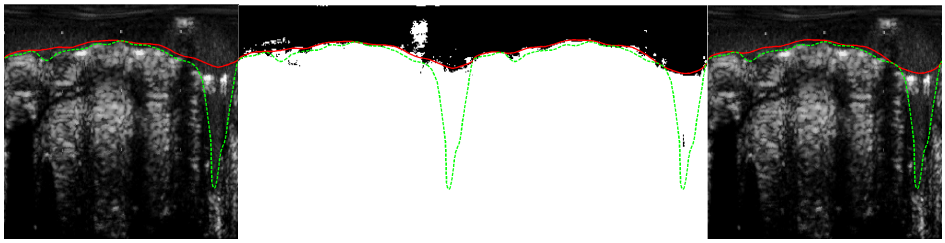
Figura 5.13: Comparativa de tiempos de ejecución 3D contra 2D.

Cualitativamente vemos en las Figuras 5.14, 5.15 y 5.16 que en el caso FP hay una mejora considerable ya que clasifica correctamente la mayor parte de zona de no lumen que antes se clasificaba como lumen. En el caso FN vemos que una zona aislada mal clasificada como no lumen ahora desaparece y obtenemos una clasificación más correcta. El caso TP no tiene diferencias mayores, aunque parece que el borde se ajusta más al manual.



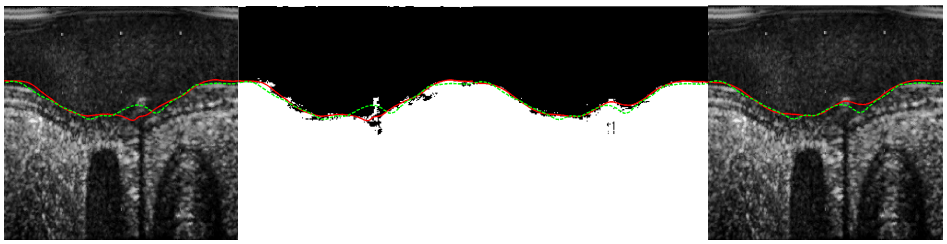
(a) (b) (c) (d)

Figura 5.14: FP MSSL 3D contra 2D, los mapas de clasificación son del segundo clasificador. (a) imagen original segmentada con 3D; (b) mapa versión 3D; (c) mapa versión 2D; (d) imagen segmentada con la configuración 2D.



(a) (b) (c) (d)

Figura 5.15: FN MSSL 3D contra 2D, los mapas de clasificación son del segundo clasificador. (a) imagen original segmentada con 3D; (b) mapa versión 3D; (c) mapa versión 2D; (d) imagen segmentada con la configuración 2D.

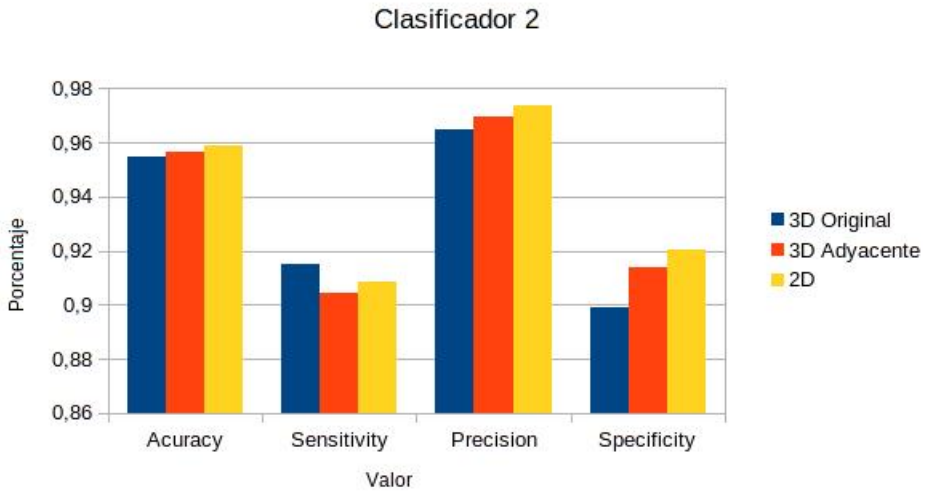


(a) (b) (c) (d)

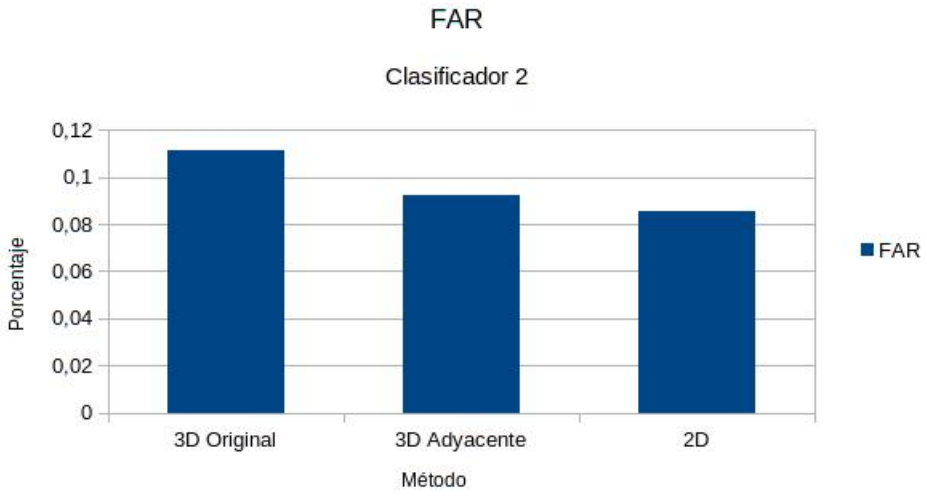
Figura 5.16: TP MSSL 3D contra 2D, los mapas de clasificación son del segundo clasificador. (a) imagen original segmentada con 3D; (b) mapa versión 3D; (c) mapa versión 2D; (d) imagen segmentada con la configuración 2D.

Finalmente decir que, las pruebas con escalas menores de 5 no han dado buen resultado con ninguno de los tres métodos, hasta el punto en que no vale

la pena compararlos con los ya mostrados. Los resultados dejan claro que **el método 2D es el más adaptado a nuestro problema**. En los gráficos de la Figura 5.17 podemos ver mas claramente la comparativa de estos tres métodos.



(a)



(b)

Figura 5.17: Gráfico con la comparativa de valores machine learning 3D Original, Adyacente y 2D.

5.3. Bifurcaciones

Para comprobar si hay mejora en la detección de frames con bifurcación hemos realizado dos pruebas. La primera ha sido añadiendo al conjunto de 8 pullbacks que tenemos uno nuevo con frames únicamente con bifurcaciones, la segunda la hemos realizado entrenando el clasificador sólo con el pullback de bifurcaciones para asegurarnos de que utiliza suficientes muestras de bifurcaciones.
Como el pullback de bifurcaciones es un conjunto de frames de distintos pacientes no hay relación entre ellos, por lo tanto hemos testeado con la configuración MSSL 2D 5 escalas ya que no tiene en cuenta la dimensión temporal, en cuanto a las features hemos escogido la configuración de las ocho mejores ya que es la que ha dado mejor resultado. Compararemos contra la misma configuración sin el pullback adicional.

Conjunto de test inicial más pullback con bifurcaciones

	Original	Bifurcaciones
%	mean±std	mean±std
H1		
Accuracy	94,22±2,48	94,05±2,61
Sensitivity	92,98±5,38	92,82±5,43
Precision	94,84±2,76	94,69±2,81
Specificity	84,79±9,93	84,58±9,60
FAR	18,44±15,32	18,59±14,91
H2		
Accuracy	96,61±1,84	96,31±2,21
Sensitivity	92,37±5,94	91,58±6,94
Precision	97,93±1,93	97,62±2,65
Specificity	93,48±7,28	93,33±6,81
FAR*	7,13±9,06	7,10±8,09

(a)

	Original	Bifurcaciones
	mean±std	mean±std
JACC(%)	85,02±9,26	85,08±9,01
HD(mm)	0,47±0,46	0,48±0,51
AE(mm^2)	0,96±0,99	1,02±1,08
PAD(%)	9,7471±8,19	10,27±8,41
MRD(mm)	0,13±0,07	0,13±0,09

(b)

Figura 5.18: Comparativa conjunto de test original contra conjunto de test original más bifurcaciones; (a) valores de *machine learning*; (b) valores del borde.

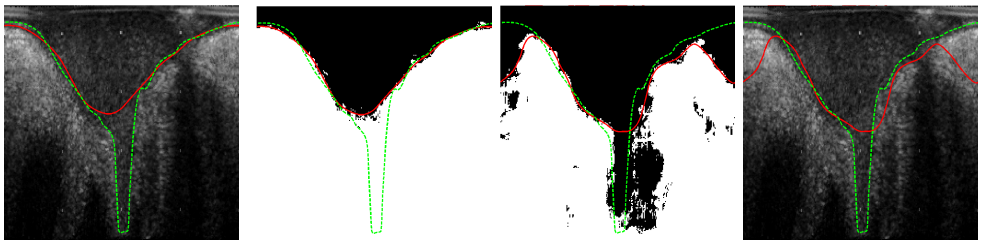
Como podemos ver en la tabla (Figura 5.18), tanto los resultados de los clasificadores como los del borde son prácticamente iguales. No hay ninguna diferencia cuantitativa al añadir más casos de bifurcaciones a la fase de training.

Sólo pullback de bifurcaciones

	Original	Bifurcaciones
%	mean±std	mean±std
H1		
Accuracy*	94,22±2,48	91,15±5,12
Sensitivity*	92,98±5,38	91,92±4,37
Precision*	94,84±2,76	90,54±7,29
Specificity*	84,79±9,93	79,02±13,23
FAR*	18,44±15,32	28,45±24,51
H2		
Accuracy*	96,61±1,84	91,05±5,61
Sensitivity*	92,37±5,94	83,69±17,56
Precision*	97,93±1,93	91,92±9,65
Specificity*	93,48±7,28	85,02±14,67
FAR	7,13±9,06	20,58±26,89

Figura 5.19: Comparativa conjunto de test original contra conjunto de test sólo de bifurcaciones, valores de *machine learning*.

No hace falta observar los valores del borde ya que a simple vista vemos que entrenando sólo con frames con bifurcaciones el resultado es pésimo, esto es debido a que estamos clasificando frames con todo tipo de dificultades entrenando el clasificador sólo para un caso específico. Lo que nos interesa es ver cualitativamente si ahora mejora la detección de las bifurcaciones que antes no clasificaba correctamente.
Si miramos los mapas del segundo clasificador de un frame con bifurcación en la Figura 5.20, observamos que ahora sí detecta la bifurcación aunque el resto de la clasificación sea bastante peor. Existe otro problema, la curva *snake* no consigue ajustarse a la bifurcación, por lo tanto habría que intentar flexibilizar la curva sin perder precisión en los demás frames sin bifurcaciones.
En resumen, el problema de detección de bifurcaciones quizá sea solucionable si encontramos una configuración de conjunto de test que contenga suficientes muestras de bifurcaciones y un nuevo ajuste de los parámetros del *snake*. Aunque al no mejorar al añadir al conjunto de test casos de bifurcaciones, parece difícil que este caso pueda segmentarse correctamente juntamente con los casos sin bifurcación. La mejor solución aparente es entrenar un clasificador específico para bifurcaciones y segmentar estos casos con él, como se hizo en el trabajo de [9].



(a) (b) (c) (d)

Figura 5.20: TP training con pullbacks originales contra trainging sólo con bifurcaciones, los mapas de clasificación son del segundo clasificador. (a) imagen pullbacks originales; (b) mapa de clasificación pullbacks originales; (c) mapa de clasificación pullbacks de bifurcaciones; (d) imagen segmentada pullbacks de bifurcaciones.

6. Conclusión

Llegados a este punto vamos a resumir en lo que hemos logrado una mejora significativa (y en lo que no) con todos estos cambios y que hemos comprobado con las pruebas pertinentes.

En cuanto a las features hemos añadido dos nuevos filtros, y hemos detectado las features que menos relevancia tenían. Ahora tenemos una configuración de ocho características que incluyen estos dos nuevos filtros, y que dan un mejor rendimiento que la configuración inicial. Todo esto sin aumentar el número de features y por lo tanto sin aumentar la cantidad de datos a procesar.

Por otra parte hemos comprobado que una versión 3D de MSSL no nos da ninguna ventaja sobre la versión 2D, es más, empeora los resultados en cuanto a la clasificación. Pero lo más importante es que hemos reducido la complejidad del sistema y los tiempos de ejecución radicalmente, obteniendo una disminución del tiempo de training en un 90 % y de test en un 65 %. Por lo referente al número de escalas de MSSL hemos visto que reducirlas a un valor menor que 5 empeora claramente los resultados de la segmentación.

El problema de las bifurcaciones es un tema complejo que no hemos podido solucionar, aunque hemos dilucidado alguna posible solución que se comentará en la siguiente sección.

Todo esto nos lleva a determinar que la mejor configuración obtenida es la siguiente:

- **Features:** ocho mejores features.
- **MSSL:** MSSL 2D a 5 escalas.
- **Conjunto de test:** ocho pullbacks iniciales.

A continuación mostraremos los datos de la comparación de la configuración original contra esta nueva para corroborar la hipótesis.

	Original	Nueva	Ganancia
%	mean±std	mean±std	
H1			
Accuracy*	93,47±3,03	94,22±2,48	0,75
Sensitivity	92,60±5,64	92,98±5,38	0,38
Precision*	94,04±3,70	94,84±2,76	0,79
Specificity	82,92±11,57	84,79±9,93	1,87
FAR	21,71±19,10	18,44±15,32	-3.26
H2			
Accuracy*	95,49±2,40	96,61±1,84	1,12
Sensitivity	91,48±6,48	92,37±5,94	0,89
Precision*	96,45±3,13	97,93±1,93	1,48
Specificity*	89,88±8,80	93,48±7,28	3.60
FAR*	11,13±10,74	7,13±9,06	-4,00

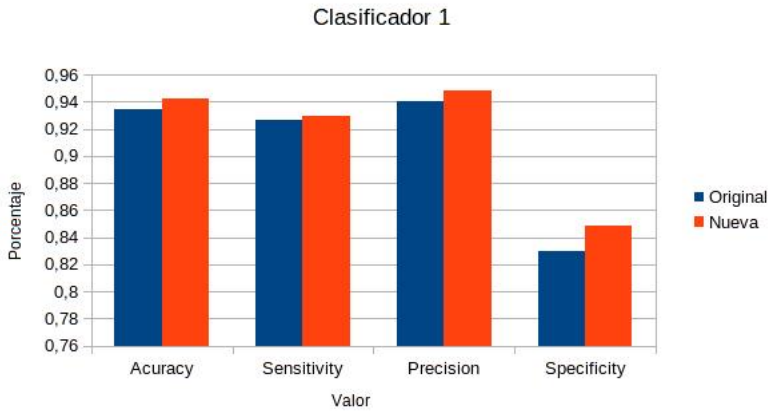
(a)

	Original	Nueva	Ganancia
	mean±std	mean±std	
JACC(%)*	81,29±10,96	85,02±9,26	3,72
HD(mm)*	0,60±0,55	0.47±0.46	-0.13
AE(mm^2)	1,10±1,13	0.96±0.99	-0,14
PAD(%)	10,95±8,915	9,74±8,19	-1,20
MRD(mm)*	0,17±0,11	0.13±0.07	-0,04

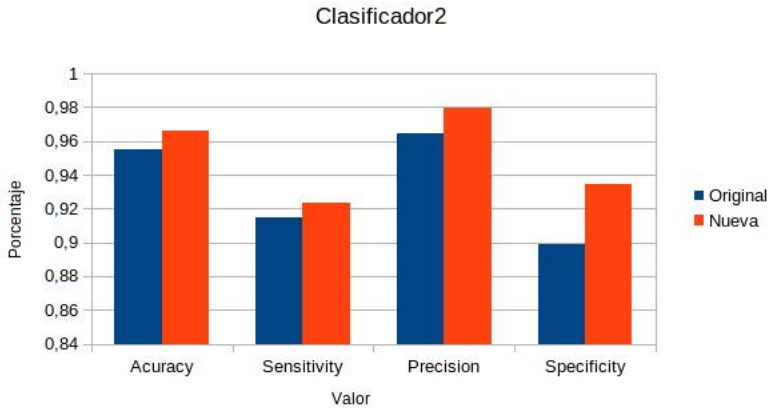
(b)

Figura 6.1: Comparativa configuración inicial contra final; (a) valores *de machine learning*; (b) valores del borde.

Como se ve a simple vista, cuantitativamente la nueva configuración mejora absolutamente todos los valores. Si nos fijamos en los resultados de machine learning los cambios más destacados se encuentran en ***specificity*** y **FAR** del segundo clasificador. En cuanto a los resultados del borde, el índice Jaccard (**JACC**) y el porcentaje de la diferencia de área (**PAD**) son los valores con una ganancia más clara. Sin duda lo que hemos conseguido con esta nueva configuración es rebajar la tasa de falsos positivos, en las figuras siguientes podemos observar gráficos que muestran esta comparativa de manera más clara.

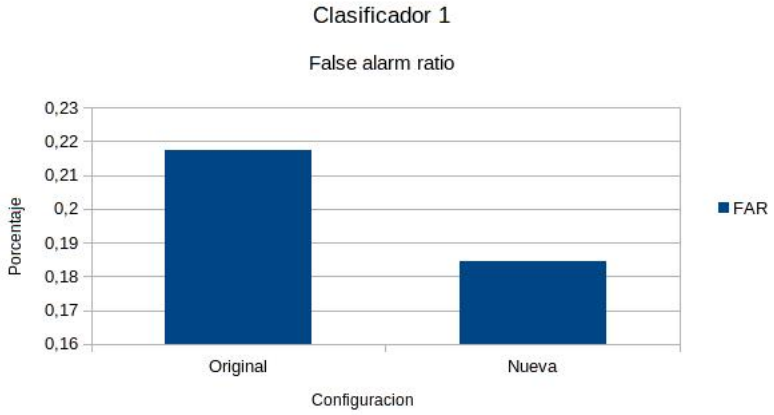


(a)

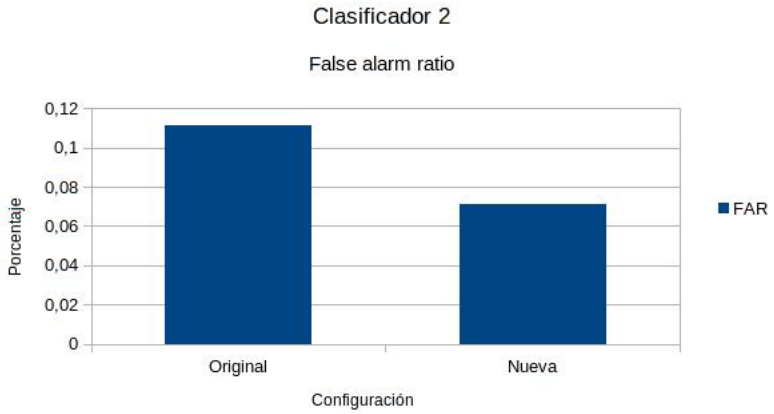


(b)

Figura 6.2: Gráficas comparativas configuración original contra nueva, valores de *machine learning*; (a) clasificador 1; (b) clasificador 2.



(a)



(b)

Figura 6.3: Gráficas comparativas configuración original contra nueva, valores *false alarm ratio* (a) clasificador 1; (b) clasificador 2.

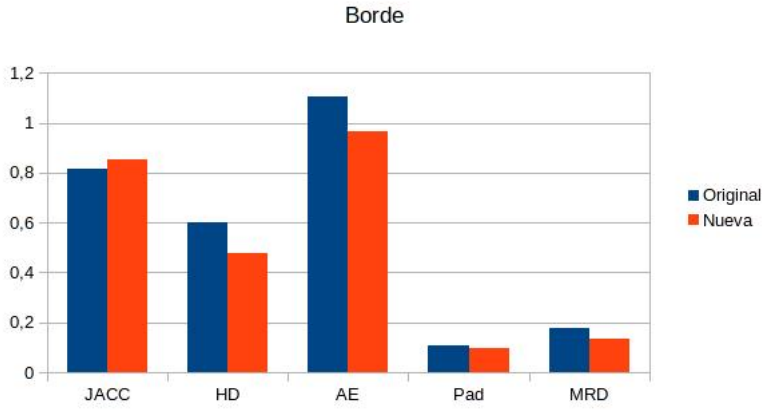
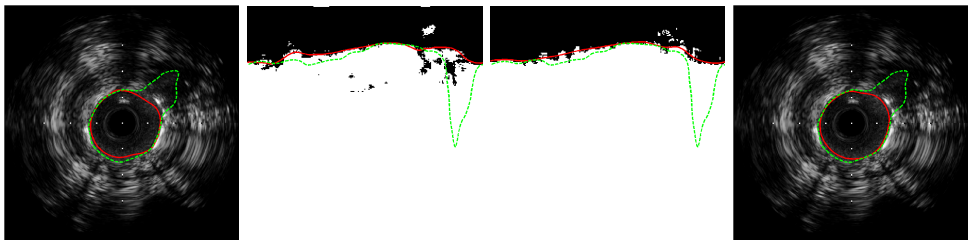


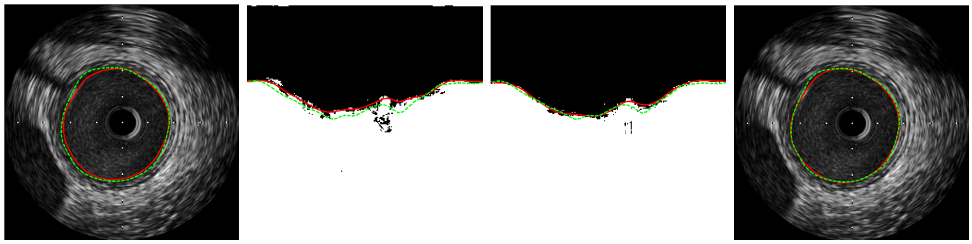
Figura 6.4: Gráfica comparativa configuración original contra nueva, valores del borde

Veamos ahora cualitativamente cuales son los resultados en algunos de los frames críticos. Mostraremos las imágenes segmentadas cartesianas ya que son más intuitivas que las polares, junto con estas se muestran los mapas del segundo clasificador (en polar).



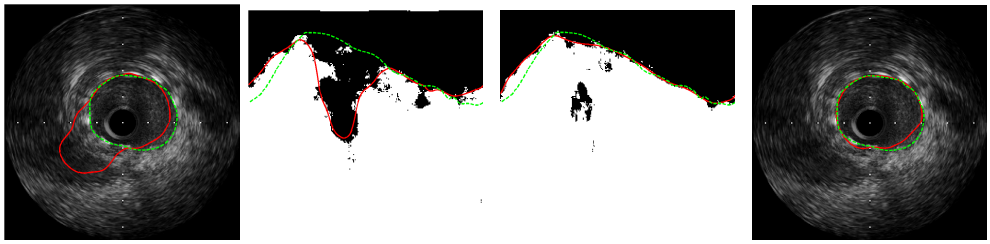
(a) (b) (c) (d)

Figura 6.5: FN configuración original contra la nueva, los mapas de clasificación son del segundo clasificador. (a) imagen original segmentada con configuración original; (b) mapa configuración original; (c) mapa configuración nueva; (d) imagen original segmentada con la nueva configuración.



(a) (b) (c) (d)

Figura 6.6: TP configuración original contra la nueva, los mapas de clasificación son del segundo clasificador. (a) imagen original segmentada con configuración original; (b) mapa configuración original; (c) mapa configuración nueva; (d) imagen original segmentada con la nueva configuración.



(a) (b) (c) (d)

Figura 6.7: FP configuración original contra la nueva, los mapas de clasificación son del segundo clasificador. (a) imagen original segmentada con configuración original; (b) mapa configuración original; (c) mapa configuración nueva; (d) imagen original segmentada con la nueva configuración.

Como vemos en la Figura 6.5 no hemos conseguido mejorar en la detección de las bifurcaciones, el caso más claro de **falso negativo**. El mapa binario de la clasificación parece más suavizado que con la configuración anterior pero sigue sin clasificar bien la bifurcación.
En el caso del ejemplo de ***true positive*** (Figura 6.6), vemos que hay una leve mejora, el mapa de clasificación es más preciso y la segmentación final casi se ajusta a la perfección a la manual. Por lo tanto obtenemos una mejora leve en estos casos.
El gran cambio viene en los casos de **falso positivo** (Figura 6.7), aquí se ve reflejada la bajada considerable del valor ***false alarm ratio***. El mapa de clasificación con la nueva configuración es claramente mejor que en la configuración anterior, clasifica correctamente toda la zona de sombra que antes era clasificada de manera incorrecta como lumen. Los falsos positivos en general son los casos en los que hemos mejorado drásticamente.

Como conclusión determinamos que con esta nueva configuración obtenemos mejores resultados en todos los aspectos, exceptuando en la detección de bifurcaciones (caso que no ha mejorado ni empeorado), por lo tanto tomamos esta configuración final como válida.

7. Trabajo futuro

En esta sección exponemos posibles vías de estudio para solucionar problemas que aún están pendientes o intentar incrementar el rendimiento de este sistema de clasificación.

El problema de la detección de las bifurcaciones es más complejo de lo que parecía inicialmente. Como hemos visto en la sección 5.3, no hemos conseguido que el clasificador detecte las bifurcaciones al tratar todos los posibles casos a la vez (presencia o no de stent, calcificaciones, bifurcaciones, etc.). Además, dado que hemos añadido suficientes muestras de bifurcaciones al conjunto de training original sin obtener resultados parece difícil que se pueda tratar este caso conjuntamente con todas las demás dificultades. De todas maneras, habría que estudiar mejor el problema de las muestras de training antes de descartar del todo la posibilidad. Esto significa estudiar en profundidad el porcentaje de cada caso que utiliza el clasificador, y ver si existe algún equilibrio en el que se pueda segmentar correctamente frames con cualquier tipo de dificultad.
También hemos visto que al entrenar específicamente un clasificador para segmentar bifurcaciones los resultados mejoraban en cuanto a la clasificación de éstas. Por eso propongo excluir las bifurcaciones de este sistema, y utilizar otro clasificador para este caso específico.

Otra posible vía de mejora podría ser un reajuste de la curva *snake*. Observando casos de falsos positivos, he visto que, aunque la clasificación no era demasiado precisa, se podría solventar limitando la flexibilidad de la curva para que no se ajustara tanto a cambios bruscos. Esto podría hacer que mejorara en muchos casos la segmentación, pero existe un dilema. Si no vamos a excluir las bifurcaciones de este sistema, la curva en vez de ser menos flexible tendría que flexibilizarse mucho más para poder ajustarse a los mapas de clasificación de bifurcaciones. Esto es necesario ya que las bifurcaciones en mapas polares son cambios bruscos y muy estrechos horizontalmente (ver mapa binario de la Figura 6.5), pero tenemos otro problema, si flexibilizásemos la curva obtendríamos segmentaciones muy irregulares y perderíamos precisión.
Este es un tema complicado que necesita un estudio en profundidad que no he podido realizar en este trabajo por problemas de tiempo.

Finalmente, existe otro cambio posible que debería comprobarse. La idea es implementar una versión MSSL 3D que trabaje sobre los frames inmediatamente adyacentes, es decir, ahora para crear el conjunto extendido de features se utilizan los frames gated adyacentes, la idea es no utilizar los frames gated sino los del pullback original. Esto podría mejorar el rendimiento de la versión 3D de MSSL ya que los frames cercanos del pullback original tienen más relación entre ellos que los gated.

Espero que estas ideas sirvan para las personas que sigan trabajando en este sistema.

8. Anexo

8.1. Estructura del Código

En este apartado comentamos la estructura general del código del sistema de segmentación de lumen. Cabe destacar que no pretende ser una documentación exhaustiva del código, sino una guía para no perderse entre las múltiples funciones, archivos y carpetas del proyecto. Comentaremos de qué manera hay que disponer las imágenes y *labels* de entrada en las carpetas y los nombres de archivo que deben tener. También explicaremos qué hace cada archivo ".m" y para qué sirve cada función.
El proyecto ha sido refactorizado, se han eliminado todas las funciones que ya no se utilizan, archivos duplicados, eliminado código comentado y reescrito algunas partes para mejorar su comprensión y la aplicación de cambios.

El código se divide en dos carpetas principales, la primera carpeta llamada "TRAINTEST" contiene el código que entrena el clasificador y testea utilizando el método *leave one patient out*. La segunda carpeta llamada "TEST" contiene la parte del sistema que sólo ejecuta la fase de test sobre un conjunto de frames y los segmenta. Esta última parte también puede necesitar los *labels* manuales para evaluar el rendimiento del clasificador después de la segmentación. Dado que TEST es un subconjunto del código TRAINTEST y tiene la misma estructura sólo comentaremos este último. Hay que aclarar que lo que se incluye en la entrega del trabajo no incluye datos de entrada (ni frames ni *labels*), ya que ocupan demasiado.

Train y test

Veamos primero la estructura de carpetas y archivos de la Figura 8.1.

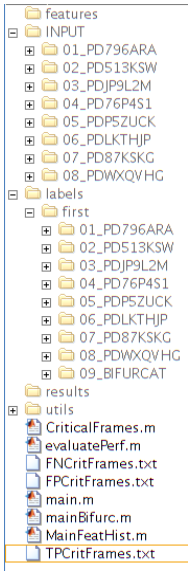


Figura 8.1: Jerarquía general de carpetas del proyecto.

Ésta es la descripción de cada carpeta:

- **features:** en esta carpeta se almacenarán archivos ".mat" que guardan las features extraídas de todos los frames de cada pullback, un archivo por cada pullback.
- **INPUT:** es la carpeta que contiene las imágenes de entrada a segmentar, vemos que hay subcarpetas, una por cada pullback. Dentro de cada subcarpeta están las imágenes cartesianas de tamaño 512 x 512 px, y que se nombran de la siguiente manera: "frame_xxxx_yyy.png", dónde "xxxx" es el número de frame (que deben ser consecutivos) y "yyy" es 002, 003 o 004 que indica si es el frame anterior, central o posterior. Por lo tanto por cada imágen xxxx tendremos tres que corresponden a la anterior adyacente, a la central y a la posterior adyacente, son necesarias para el filtro de correlación de Pearson.
- **labels:** contiene una subcarpeta "first" en la que se almacenan archivos ".mat" con la segmentación manual, y tiene una carpeta por cada pullback. El nombre de estos archivos es del tipo: "PULLBACK_xxxx_frame_y.mat", donde "PULLBACK" es el nombre del pullback, "xxxx" es el número de frame que tenía en el pullback original y "y" es el número de frame que corresponde con la imágen de la carpeta INPUT. Estos archivos deben

contener una matriz llamada "YYadv" con las coordenadas del borde manual.

- **results:** dentro de esta carpeta se almacenarán los resultados de la clasificación una vez ejecutado el algoritmo. Esto comprende clasificadores fuertes, imágenes segmentadas, mapas de clasificación, archivos de texto con la evaluación de rendimiento, etc.

- **utils:** contiene una jerarquía de carpetas donde se almacenan archivos de código Matlab necesarios. Comentaremos su contenido más adelante.

A continuación describimos los archivos que hay en la raíz del proyecto:

- **CriticalFrames.m:** es un script escrito por mi que extrae las imágenes y los mapas de clasificación de los frames indicados en los archivos "**FNCriticalFrames.txt**", "**FPCriticalFrames.txt**" y "**TPCriticalFrames.txt**" y las guarda en una única imagen en la carpeta results para facilitar la evaluación de la segmentación. En los archivos de texto hay una lista de números en dos columnas, la columna de la derecha es el número del pullback al que pertenece el frame y la de la izquierda es el frame que queremos extraer.

- **main.m:** es el script principal que se encarga de llamar a funciones para preparar los datos, extraer features, realizar el training y test del clasificador y evaluar el rendimiento.

- **mainBifurc.m:** es básicamente el mismo script que "main.m" con modificaciones para entrenar el clasificador sólo con un pullback de bifurcaciones, el nombre de este pullback (que no se incluye en la entrega del código por razones de tamaño) debe empezar por "09".

- **MainFeatHist.m:** este script escrito por mí genera un histograma con las N features más relevantes a partir de los clasificadores creados una vez se ha ejecutado todo el algoritmo. El histograma es útil para comprobar la relevancia de las caracterísitcas iniciales con las que se entrena el primer clasificador.

Todas las funciones importantes están en la carpeta "**utils**", veamos qué contiene en la Figura 8.2.

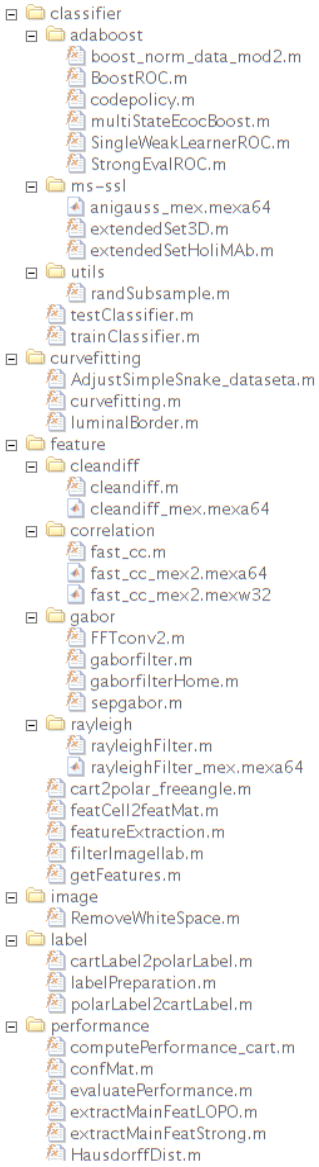


Figura 8.2: Contenido de la carpeta utils del código.

He intentado agrupar los archivos Matlab en carpetas por "temas" de una manera coherente para que se más fácil encontrarlos. Así tenemos por ejemplo una carpeta "classifier" que contiene las funciones referentes al clasificador, adaboost, etc.
Ahora enumeraremos las carpetas/archivos, y describiremos su función.

Classifier

- **adaboost:** contiene una serie de funciones para entrenar el clasificador o clasificar frames. En concreto "**boost_norm_data_mod2.m**" aplica adaboost y devuelve un clasificador fuerte, y "**multiStateEcocBoost.m**" aplica un clasificador sobre un frame y devuelve un mapa de probabilidad de clasificación. El resto son funciones de soporte para estas dos.

- **ms-ssl:** contiene dos funciones para la extracción de un conjunto extendido de features con MSSL. En concreto "**extendedSet3D.m"** se encarga de la extracción de features en tres dimensiones, y "**extendedSetHoliMAb.m**" extrae las features contextuales en dos dimensiones, todo esto a partir de los mapas de probabilidad del primer clasificador. Las dos funciones devuelven una matriz con las características extendidas de cada píxel para todos los frames.

- **utils:** únicamente guarda la función "**randSubsample.m**", que es la que se encarga de escoger aleatoriamente un subconjunto de X muestras (píxeles) del conjunto total de training para utilizarlas en el clasificador.

- **classifier:** en la raíz de esta carpeta hay dos funciones, la primera es "**testClassifier.m**" que se encarga de realizar la clasificación de un conjunto de frames utilizando un clasificador ya entrenado, devuelve un conjunto de mapas binarios de clasificación para cada frame. La segunda es "**trainClassifier.m**", entrena un clasificador fuerte a partir de un conjunto de training, devuelve dos clasificadores fuertes correspondientes a la primera y segunda fase de clasificación.

Curvefitting

- **luminalBorder.m:** es un script que se encarga de aplicar la última parte del sistema a los mapas binarios de clasificación. Por cada mapa binario genera el borde (curva *snake*) utilizando la función "**curvefitting.m**", muestra los frames con la segmentación final y los mapas de clasificación por pantalla. También calcula los resultados del borde (índice Jaccard, MRD, etc.) y los almacena en la carpeta "results".

Feature

- **cleandiff:** en esta carpeta se encuentra la función "**cleandiff.m**", escrita por mi, que realiza el filtrado de la imagen con la feature de diferencia de nivel de gris ponderada. También incluye un archivo "mex" de este mismo filtro compilado para Linux 64 bits.

- **correlation:** contiene dos archivos "mex" que filtran la imágen con la correlación de Pearson, hay una versión compilada para Linux 64 bits y otra para Windows 64 bits. El archivo "**fast_cc.mex**" únicamente prepara los datos para realizar el filtrado utilizando uno de los dos archivos "mex".

- **gabor:** "**gaborfilter.m**" y "**sepgabor.m**" son funciones que aplican un filtro de gabor a la imágen, las dos tienen la misma función. Las demás funciones de esta carpeta son funciones de apoyo para estas dos.
- **rayleigh:** "**rayleighFilter.m**" es el filtro Rayleigh que he escrito para este trabajo, el archivo "mex" es la misma función compilada para Linux 64 bits.
- **cart2polar_freeangle.m**: realiza la conversión de una imagen de coordenadas cartesianas a polares.
- **FeatCell2FeatMat.m:** es una función de apoyo que convierte el *cell array* de features generado por el algoritmo a una matriz para facilitar el uso de esta información en otras partes del código.
- **featureExtraction.m:** este script se encarga de extraer las features de cada frame y las almacena en archivos ".mat" en la carpeta "features" de la raíz.
- **filterImageIlab.m:** elimina artifacts de las imágenes de tipo DICOM IVUS para su posterior uso.
- **getFeatures.m:** es la función que dada una imagen, la filtra con el conjunto de filtros de extracción de features y devuelve el resultado en una matriz de tamaño [#píxeles x #features]. En esta función es donde añadimos o eliminamos filtros.

Image

- **RemoveWhiteSpace.m:** función de utilidad para eliminar la región de color blanco alrededor de una imagen.

Label

- **cartLabel2polarLabel.m:** convierte las coordenadas de un *label* manual de cartesianas a polares.
- **polarLabel2cartLabel.m:** convierte las coordenadas de un *label* manual de polares a cartesianas.
- **labelPreparation.m:** prepara los datos de los *labels* manuales para su utilización en el clasificador. En concreto los convierte a coordenadas polares y los agrupa en una única matriz por cada pullback que se almacena en la carpeta "labels/final" de la raíz del proyecto.

Performance

- **computePerformance_cart.m:** calcula los datos de rendimiento del borde (índice Jaccard, Haussdorf Distance, etc.) a partir de la segmentación automática y la manual.

- **confMat.m:** calcula la matriz de confusión, es la matriz que dice cuantos casos hay de FP, FN, TP y TN.
- **evaluatePerformance.m:** calcula los valores de rendimiento del clasificador (FAR, accuracy, etc.) para los dos clasificadores.
- **extractMainFeatLOPO.m:** genera, a partir de todos los clasificadores creados en la fase de training y test con LOPO, un histograma con las N mejores features que muestra cuantas veces se utilizan. Usa la función "**extractMainFeatStrong.m**".
- **HaussDorffDist.m:** calcula la distancia de Hausdorrf entre dos bordes de segmentación.

8.2. Conversión Automática de Código Matlab a C

A partir de la versión 2012 de Matlab, se incluye una herramienta muy útil para la generación automática de código C a partir de código Matlab y la compilación en archivos "mex". En este trabajo nos ha sido muy útil para compilar las funciones de las nuevas features añadidas y agilizar su ejecución.
Vamos a exponer un pequeño manual para compilar funciones Matlab a archivos mex, utilizando como ejemplo el filtro Rayleigh.

Para poder compilar una función debemos cumplir unos requisitos:

1. En el código no puede aparecer ningún warning de ningún tipo, por lo tanto debemos hacer los cambios necesarios para eliminarlos y que arriba a la derecha, en la ventana del código, nos aparezca una marca en verde que significa que no hay ningún warning.
2. Una vez eliminados los warnings, justo después de la cabecera de la función debemos escribir estos comandos:

   ```
   %#codegen
   coder.inline('never')
   ```

3. Otro aspecto a tener en cuenta es que, si estamos usando funciones de Matlab en nuestras funciones, hay que saber que no todas son compilables. Por ejemplo, en mi función "rayleighFilter" utilizo una función llamada "raylfit" que no es compilable (esto lo sabemos cuando intentamos generar el archivo "mex"). para solucionar esto busqué el archivo "raylfit.m" dentro de las carpetas de instalación de matlab, apliqué los pasos 1 y 2, y copié el archivo con un nombre distinto en la misma carpeta donde se encontraba "rayleigFilter.m". De esta manera pude compilar el filtro sin problemas.

Una vez tenemos todo el código preparado pasamos a compilarlo.
Para ello lo abrimos la función en Matlab y escribimos en la consola el comando "coder" y pulsamos enter. Se nos abrirá una ventana como la de la Figura 8.3.

Seleccionamos donde queremos guardar el proyecto que generaremos y hacemos click en “OK”.

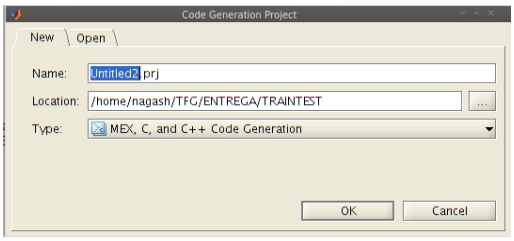


Figura 8.3: Ventana de opciones coder.

Hecho esto veremos la ventana de la Figura 8.4. En la pestaña “Overview”, “Entry-point files” hacemos click en “Add Files” y seleccionamos el archivo “.m” donde está la función del filtro. Esto detectará cuales son los parámetros de entrada para los que deberemos indicar el tipo de datos (int, double, etc.) y el tamaño de estos datos de entrada, que puede ser fijo o variable.

Figura 8.4: Ventana de generación de código.

Para definir los tamaños hacemos click con el botón derecho encima de uno de ellos y en el menú que aparece seleccionamos la opción “Define Type...”. Se abrirá otra ventana (Figura 8.5, ventana izquierda) con opciones para definir el tipo de dato y el tamaño, si no queremos un tamaño fijo clickamos en la ruedecilla de la derecha y seleccionamos “Edit Size Vector Definition...”. Esto nos permitirá definir tamaños variables con o sin un límite superior (Figura 8.5, ventana derecha).

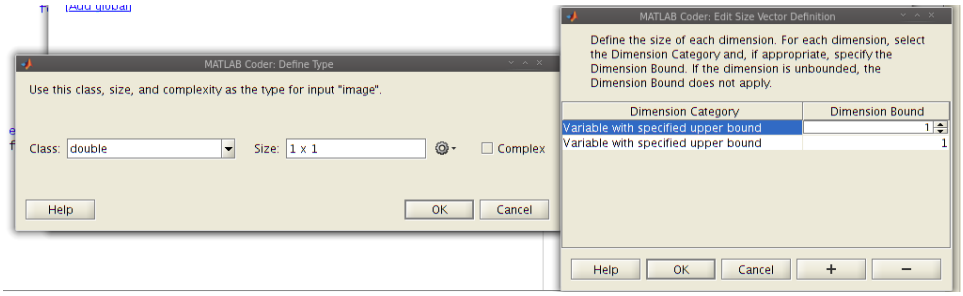


Figura 8.5: Ventana de tipo de variable y tamaño.

Una vez hemos definido el tipo de datos de todos los parámetros de entrada vamos a la pestaña "Build" (Figura 8.6), indicamos el nombre del archivo "mex" de salida y en "Output type" seleccionamos "MEX Function".

Figura 8.6: Ventana "Build".

Finalmente, clickamos en "Build" y se generará una carpeta con el código Matlab convertido a lenguaje C y un archivo "mex" que podemos utilizar desde Matlab.

9. Bibliografía

Referencias

[1] Gatta, Carlo and Puertas, Eloi and Pujol, Oriol. "Multi-scale stacked sequential learning". Pattern Recognition, 44 (10-11) :2414–2426, 2011

[2] Francesco Ciompi, Oriol Pujol, Carlo Gatta, Marina Alberti, Simone Balocco, Xavier Carrillo, et al. (2012). HoliMab: A Holistic Approach for Media-Adventitia Border Detection in Intravascular Ultrasound. MIA, 16(6), 1085–1100

[3] Marina Alberti. Detection and Alignment of Vascular Structures in Intravascular Ultrasound using Pattern Recognition Techniques, B. 8493-2013, (2012).

[4] R. C. Chivers. "The scattering of ultrasound by human tissues, some theoretical models." Ultrasound in Medicine and Biology, pp. 1-13, (1977).

[5] R. F. Wagner, M. F. Insana and D. G. Brown. "Unified approach to the detection and classification of speckle texture in diagnostic ultrasound." Optical Engineering, pp. 738-742, (1986).

[6] https://www.cvc.uab.es/IVUSchallenge2011/

[7] O. Pujol, D. Gil, P. Radeva, Fundamentals of STOP and Go Active Models, Image and Vision Computing, Volume 23, Issue 8, 1 August 2005, Pages 681-691, (ISI 1,159).

[8] Marie-Hélène Roy Cardinal, Jean Meunier, Gilles Soulez, Roch L. Maurice, Éric Therasse, and Guy Cloutier* "Intravascular Ultrasound Image Segmentation: A Three-Dimensional Fast-Marching Method Based on Gray Level Distributions" (2006)

[9] M. Alberti, S. Balocco, C. Gatta et al., Automatic Bifurcation Detection in Coronary IVUS Sequences, IEEE Trans. Biomed. Engineering 59(4): 1022-1031 (2012)

[10] http://en.wikipedia.org/wiki/Rayleigh_distribution

[11] http://en.wikipedia.org/wiki/Wilcoxon_signed-rank_test

[12] http://en.wikipedia.org/wiki/AdaBoost

[13] http://www.mathworks.es/products/matlab-coder/

Índice de figuras