Proyecto Fin de Carrera

Ingeniería de Telecomunicación

Tratamiento digital de imágenes aplicado a lesiones pigmentadas de la piel

Autor: Samir Darouich

Tutor: María del Carmen Serrano Gotarredona

Dep. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Proyecto Fin de Carrera Ingeniería de Telecomunicación

Autor:

Samir Darouich

Tutor:

María del Carmen Serrano Gotarredona

Dep. de Teoría de la Señal y Comunicaciones

Escuela Técnica Superior de Ingeniería

Universidad de Sevilla

Sevilla, 2016

Proyecto Fin de Carrera: Formato de Publicación de la Escuela Técnica Superior de Ingeniería de Sevilla

Autor: Samir Darouich

Tutor: María del Carmen Serrano Gotarredona

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2016

El Secretario del Tribunal

A mi familia

A mis amigos

Agradecimientos

En primer lugar quiero agradecerle a mi familia el apoyo que me han dado desde el momento en el que decidí emprender mi trayectoria universitaria. Gracias a las condiciones que me han ofrecido he podido cumplir mi sueño. Tanto mis padres como mis dos hermanos han estado siempre aquí en lo bueno y en lo malo, ayudándome a superar aquellos momentos difíciles y a hacer más llevadera mi carrera universitaria. Una mención especial a mi padre, que a pesar de que ya no está entre nosotros, estaría muy orgulloso de mí en este día tan importante.

Gracias a mi madre, por los sacrificios que ha empleado para que llegue a esta fase, por su incansable apoyo y paciencia durante todos estos años. Gracias a mi hermano y hermana por su inagotable apoyo.

Mi paso por la Escuela Superior de Ingenieros de la Universidad de Sevilla no ha sido sólo una mera formación académica, sino que me ha servido para formarme también a nivel personal, adquiriendo nuevas experiencias y conociendo a gente que hoy en día forman parte de mi vida.

Gracias a mis amigos y compañeros de la carrera que han hecho más llevaderos todos estos años.

También me gustaría agradecerle a mi tutora María del Carmen Serrano Gotarredona por el apoyo que me ha ofrecido durante estos meses para la realización de este proyecto, por sus concejos, su entrega y disponibilidad. Agradecerle también sus enseñanzas no solo durante el desarrollo del proyecto sino también como muy buena profesora durante la carrera.

Por todo esto y sin todos ellos nunca habría sido posible.

Gracias a todos os quiero.

Samir Darouich

Resumen

El presente proyecto de fin de carrera trata de desarrollar una máquina de aprendizaje para la detección automática de regiones de velo azul-blanco en imágenes dermatoscópicas de lesiones pigmentadas de la piel, que es una característica relevante del melanoma. El proyecto se basa sobre el artículo *Automatic detection of blue-white veil and related structures in dermoscopy images de M.Emre Celebi, Hitoshi Iyatomi, William V.Stoecker, Randy H.Moss, Harold S.Rabinovitz, Giuseppe Argenziano, H.Peter Soyer*.

Para la implementación del algoritmo, se utilizó el software matemático Matlab que gracias a sus herramientas de tratamiento digital de imágenes, nos facilitó el desarrollo completo del proyecto.

En lo que es el método usado, se compone de varias etapas que incluyen pre-procesado, extracción de características de color, la inducción de un árbol de decisión, inducido por el conocido algoritmo C4.5, la aplicación de las reglas del árbol sobre un conjunto de 90 imágenes dermatoscópicas. Y finalmente un post procesado para obtener las regiones de velo azul-blanco finales.

El set de 90 imágenes las obtenemos de un Atlas de dermatoscopia. La finalidad del proyecto es obtener unos buenos resultados de sensibilidad y especificidad en la detección de esas regiones de velo, e incluso intentar superar los resultados del artículo.

Abstract

In this project we present a machine learning approach to the automatic detection of blue-white veil and related structures in dermoscopy images of pigmented skin lesions, which is an important characteristic of melanoma. The project is based on the paper *Automatic detection of blue-white veil and related structures in dermoscopy images* of *M.Emre Celebi, Hitoshi Iyatomi, William V.Stoecker, Randy H.Moss, Harold S.Rabinovitz, Giuseppe Argenziano, and H.Peter Soyer.*

For the development of this project, we used the mathematical software Matlab. Its digital image processing tools, gave us the full comfort to perform the whole project

The method is comprised of several steps including pre-processing, color features extraction, decision tree induced by the well-known algorithm C4.5, the application of tree's rules over a set of 90 dermoscopy images. And finally a post processing that obtains the final blue-white veil regions

The set of 90 images were obtained from an Atlas of dermoscopy. The main purpose of this project is to obtain some good results of sensitivity and specificity in detecting these veil regions, and even try to overcome the results of the based paper.

Índice

Agradecimientos……I

Resumen……..III

Abstract……V

Índice……...VI

Índice de tablas……VIII

Índice de figuras……...X

1 Introducción……...1
1.1 Visión general...……..2
1.2 Motivación…….2
1.3 Objetivos…….3
1.4 Estructura de la memoria……4
2 Temática y estado del arte……5
2.1 Lesión pigmentada……..6
2.1.1 Lesión pigmentada maligna o melanoma maligno……...6
2.2 Dermatoscopia……8
2.2.1 Métodos existentes de diagnósticos en dermatoscopia ……....9
2.2.1.1 Regla ABCD de la dermatoscopia…….10
2.2.1.2 New7- Point Checklist……...……11
2.2.2 Velo azul-blanco en lesiones melanocíticas……...…13
3 Método……...…...……15
3.1 Descripción del set de imágenes y pre procesado……16
3.1.1 Descripción del set de imágenes……..……...16
3.1.2 Pre procesado……..18
3.1.2.1 Determinación del contorno de la lesión……19
3.1.2.2 Determinación del color de la piel de fondo……..23
3.1.2.3 Píxeles de entrenamiento y prueba…….31
3.2 Extracción de características…….34
3.2.1 Características de color absoluto……35
3.2.2 Características de color relativo……35
3.2.3 Implementación en Matlab……36
3.3 Clasificación de píxeles……..……...49
3.3.1 Algoritmo C4.5……...40
3.3.1.1 Teoría y descripción…..……40
3.3.1.1.1 Construcción de árboles de decisión……...40
3.3.1.1.2 Origen algoritmo C4.5…….41
3.3.1.1.3 Características del algoritmo…..……..42

3.3.1.1.4 Heurística……43
3.3.1.1.5 Atributos……..…43
3.3.1.1.6 Ventajas y mejoras respecto a ID3……..43
3.3.1.1.7 Sobreajuste (Overfitting)……44
3.3.1.1.8 Post poda (Post Prunning)……...45
3.3.1.1.9 Estructuras utilizadas en el C4.5…….....45
3.3.1.1.10 Pseudocódigo……...46
3.3.1.1.11 Diagrama genérico……...47
3.3.1.1.12 Estimación de la proporción de errores……...48
3.3.1.2 Implementación en Matlab…….48
3.3.2 Árbol de decisión inducido…….55
3.3.3 Detección de regiones de velo……59
4 Resultados…….66
5 Conclusiones y líneas futuras...……...……74
5.1 Conclusión personal……..……76
6 Referencias……...77

Índice de tablas

Tabla 1: Ejemplo de puntuación método New7 Point Checklist

Tabla 2: Código Matlab de la función **detec_cont.m**.

Tabla 3: Código Matlab de la función **detec_color_BG.m**.

Tabla 4: Distribución de los recortes de velo y no velo.

Tabla 5: Definición de las características de color.

Tabla 6: Código Matlab de la función **features_color_croped.m**.

Tabla 7: Código Matlab de la función **C4.5.m**.

Tabla 8: Código Matlab de la función **make_tree.m**.

Tabla 9: Código Matlab de la función **detec_reg_velo.m**.

Tabla 10: Resumen número recortes de prueba.

Tabla 11: Resultados de sensibilidad, especificidad y Vpp.

Índice de Figuras

Ilustración 1: Visión general de la máquina de aprendizaje desarrollada.

Ilustración 2: Incidencias y muertes por 100.000 personas en los EEUU entre 1992 y 2012.

Ilustración 3: Detalles de la epidermis normal y epidermis cancerígena.

Ilustración 4: Dermatoscopio convencional.

Ilustración 5: Izquierda: Melanoma (imagen clínica). Derecha: Melanoma (Imagen dermatoscópica).

Ilustración 6: Melanoma con velo azul-blanco (a) imagen clínica (b) imagen dermatoscópica.

Ilustración 7: Imágenes de melanomas con velo azul-blanco.

Ilustración 8: Imágenes de Clark nevus sin velo.

Ilustración 9: Imágenes de Dermal nevus sin velo.

Ilustración 10: Visión general del pre procesado.

Ilustración 11: Diagrama de flujos de la función **detec_cont.m**.

Ilustración 12: Imagen original con viñeta negra en los bordes.

Ilustración 13: Imagen con viñeta recortada.

Ilustración 14: Representación de la imagen en escala de grises.

Ilustración 15: Izquierda la imagen dermatoscópica, derecha aproximación del contorno de la lesión (ejemplo 1).

Ilustración 16: Izquierda la imagen dermatoscópica, derecha aproximación del contorno de la lesión (ejemplo 2).

Ilustración 17: Ejemplo cuadros negros de imagen y pelos a omitir en el cálculo del color.

Ilustración 18: Ejemplo de burbujas a omitir en el cálculo del color de piel.

Ilustración 19: Diagrama de flujos de la función **detec_color_BG.m**.

Ilustración 20: De derecha a izquierda: máscara lesión, región 1, región 2.

Ilustración 21: 10% (gris) y 20% (blanco) regiones fuera de la lesión.

Ilustración 22: Región total omitida en negro (valor lógico 0).

Ilustración 23: Ventana de la imagen con el recorte de velo seleccionado.

Ilustración 24: Imagen del recorte de la ilustración 23.

Ilustración 25: Total de los recortes guardados en fichero de datos Matlab.

Ilustración 26: Diagrama de flujos de la función **features_color_croped.m**.

Ilustración 27: Evolución del C4.5 a partir del ID3.

Ilustración 28: Estructura genérica de un árbol de decisión inducido por el C4.5.

Ilustración 29: Diagrama genérico del algoritmo C4.5

Ilustración 30: Diagrama de flujos de la función **C4_5.m**.

Ilustración 31: Trozo de la matriz PAT que contiene los atributos de entrenamiento.

Ilustración 32: Aspecto de la estructura tree nodo raíz.

Ilustración 33: Pestaña del campo “child” de la estructura tree.

Ilustración 34: Aspecto de la sub estructura tree.child (1,1).

Ilustración 35: Aspecto de las dos ramas de la sub estructura tree.child (1,1).

Ilustración 36: Aspecto de la sub estructura tree.child (1,2).

Ilustración 37: Aspecto de las dos ramas de la sub estructura tree.child (1,2).

Ilustración 38: Árbol de decisión clasificador de píxeles.

Ilustración 39: Diagrama de flujos de la función **detec_reg_velo.m**.

Ilustración 40: Ruta de la decisión Velo.

Ilustración 41: Máscara binaria inicial de regiones de velo.

Ilustración 42: Máscara binaria final de regiones de velo.

Ilustración 43: Resultado de detección de velo azul-blanco.

Ilustración 44: Árbol de decisión artículo.

Ilustración 45: Ejemplos de resultados de detección de velo azul-blanco para 8 imágenes de nuestra base de datos. Los bordes del velo están delineados con líneas blancas.

CAPÍTULO 1: Introducción

1.1 Visión general

El proyecto "Tratamiento digital de imágenes aplicado a lesiones pigmentadas de la piel" consiste en el desarrollo de una máquina de aprendizaje enfocada en la detección del velo azul-blanco y estructuras relacionadas, en imágenes dermatoscópicas de lesiones pigmentadas cutáneas. Para el desarrollo del algoritmo usado, se ha hecho uso del software matemático Matlab (R2013a).

Basándonos en el artículo *Automatic detection of blue-white veil and related structures in dermoscopy images de M.Emre Celebi, Hitoshi Iyatomi, William V.Stoecker, Randy H.Moss, Harold S.Rabinovitz, Giuseppe Argenziano, H.Peter Soyer* [1], se parte de un set de imágenes dermatoscópicas, y se les aplica una serie de operaciones de tratamiento digital de imágenes para su pre procesado, después se extraen unas características de los píxeles de color de esas imágenes, con el fin de clasificarlas mediante un árbol de decisión inducido. Dando como salida final las regiones del velo azul-blanco detectadas. La Ilustración 1 muestra una visión general del enfoque descrito.

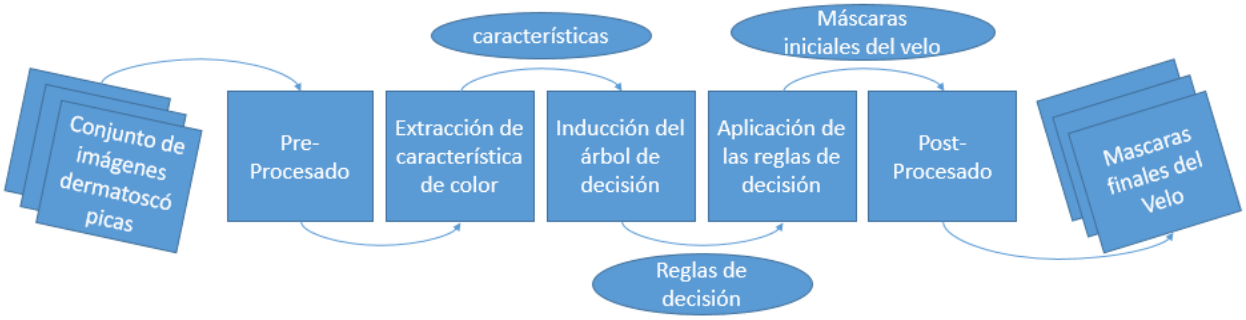


Ilustración 1. Visión general de la máquina de aprendizaje desarrollada

1.2 Motivación

Este proyecto tiene una especial importancia ya que analiza el cáncer de piel, que es la neoplasia más frecuente en el ser humano. Hay principalmente tres tipos: el carcinoma basocelular, el carcinoma espino celular y el melanoma; este último es el analizado en este proyecto, pues el melanoma maligno es el tipo de cáncer de piel más mortal que existe. También es uno de los cánceres con el aumento más rápido de casos en el mundo, con una incidencia estimada de 160.000 sólo en 2012. En España y en el mismo año se estimó un total 967 muertes [2].

La ilustración 2 muestra la evolución del número de casos nuevos y defunciones por 100.000 personas en los Estados Unidos entre 1992 y 2012, según un estudio adaptado por edad.

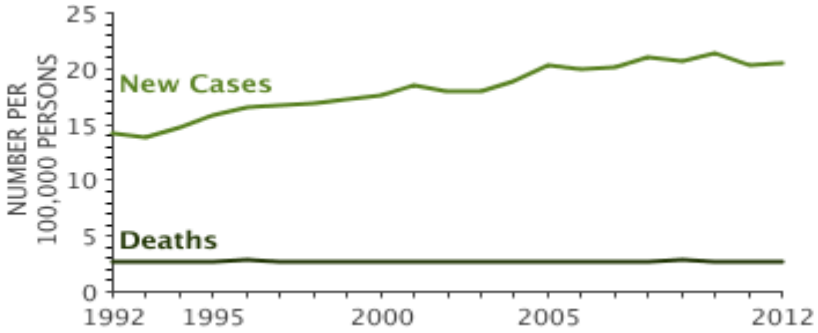


Ilustración 2. Incidencias y muertes por 100.000 personas en los EEUU entre 1992 y 2012

Fijándonos en el periodo de tiempo 2008-2012 se puede ver que el número de nuevos casos (incidencias) de melanoma fue de 21,6 por cada 100.000 hombres y mujeres por año. El número de muertes fue de 2,7 por cada 100.000 hombres y mujeres por año.

La detección precoz de esta lesión de piel aumenta las probabilidades de curación, por lo que cobra gran importancia una alternativa de bajo coste que evite la evaluación de cada paciente por parte de los dermatólogos. Dicha alternativa puede ser un sistema automático que, partiendo por ejemplo de imágenes de lesiones obtenidas de la piel con una cámara digital estándar, valore el riesgo de melanoma.

Este proyecto presenta una alternativa aún más eficiente que la descrita anteriormente ya que aplica un tratamiento digital sobre imágenes dermatoscópicas de las lesiones de piel, de manera ordenada y óptima, con el fin de diagnosticar el melanoma.

1.3 Objetivos

Este proyecto se ha desarrollado con la intención de ayudar a la detección rápida y sencilla de las lesiones melanómicas enfocándonos en uno de sus síntomas más importantes para su diagnóstico en imágenes dermatoscópicas, que es el velo azul-blanco.

Después del desarrollo completo del proyecto se va a intentar llegar a unos resultados muy óptimos en cuanto a sensibilidad (porcentaje de los píxeles de velo correctamente detectados) y especificidad (porcentaje de los píxeles de no-velo correctamente detectados):

- sensibilidad del 96.67%.
- especificidad del 97.78%.

Resultados bastante buenos y mejores que los obtenidos en el artículo en que se ha basado este PFC (En el capítulo 4 se van a analizar los resultados con más detalles).

1.4 Estructura de la memoria

El capítulo 2 introduce la temática del proyecto y el estado de arte de este campo. En dicho capítulo, se hace una definición general del cáncer de piel en lesiones pigmentadas. También se va a abordar el campo de la dermatoscopia, introduciendo la característica del velo azul-blanco en lesiones melanómicas. El capítulo en cuestión termina por un resumen de los métodos ya existentes en el diagnóstico del melanoma en el campo de la dermatoscopia.

El capítulo 3 aborda todo lo que es el desarrollo del algoritmo usado, con sus diferentes etapas:

- Descripción del conjunto de imágenes usadas.
- Pre procesado.
- Extracción de las características.
- Clasificación de píxeles.

Por otra parte, llegando al capítulo 4, presentamos los resultados obtenidos y hacemos una comparación con los resultados del artículo.

Para concluir, en el capítulo 5 se exponen las conclusiones y se argumentan las posibles líneas futuras de investigación.

CAPÍTULO 2: Temática y estado del arte

2.1 Lesión pigmentada

Las lesiones pigmentadas por lo general se refieren a las proliferaciones melanocíticas de la piel. Los melanocitos son las células productoras del pigmento (melanina) en la piel y pueden variar mucho en formas y tamaños. Las lesiones pigmentadas pueden ser benignas o malignas. Las proliferaciones benignas a menudo se llaman lunares o nevus y son las más comunes, y las proliferaciones malignas son melanomas [3].

A medida que la conciencia del cáncer de piel crece, los pacientes se están presentando a sus médicos con las primeras formas de lesiones pigmentadas. Una de las áreas más difíciles y contenciosas de la patología es el diagnóstico preciso de estas lesiones pigmentadas. Cada vez más los patólogos están llamados a hacer diagnósticos definitivos entre nevus benigno o melanoma maligno.

Cabe notar que las proliferaciones melanocíticas no son la única causa de las lesiones pigmentadas. A veces la causa es debida a un aumento de la melanina, no asociado con un aumento de los melanocitos.

Como ye se ha mencionado, la mayoría de las lesiones pigmentadas de la piel son benignos; Sin embargo, cierta preocupación ha causado que el paciente haga una investigación y una consulta médica ante lesiones pigmentadas presentando características no comunes. Pero la gran variedad de lesiones pigmentadas de la piel hace difícil que el diagnóstico sea correcto. Un diagnóstico correcto es importante, por lo que, las características físicas de la lesión, los datos demográficos del paciente, la presencia de síntomas asociados, los trastornos sistémicos relacionados, y los patrones de localización y crecimiento de la lesión, pueden dar pistas para diagnosticar y tratar adecuadamente esas lesiones. Además, para mejorar el diagnóstico, el dermatólogo debe acercarse a la evaluación de la lesión de una manera sistemática. De aquí la finalidad de este proyecto que trata de apoyar este último aspecto, desarrollando un sistema automático que ayuda a este diagnóstico [4].

Hablemos ahora del segundo tipo de lesiones pigmentadas, que nos interesa para este proyecto que es la lesión pigmentada maligna o melanoma maligno.

2.1.1 La lesión pigmentada maligna o melanoma maligno

La lesión pigmentada maligna o melanoma maligno (MM) es un tumor cutáneo maligno derivado de los melanocitos. Es uno de los tres grandes tipos de cáncer de piel que existen; los otros dos tipos son: Carcinoma de células basales, y Carcinoma de células escamosas. El MM es uno de los tumores malignos que ha experimentado un aumento de frecuencia más grande durante las últimas décadas. Suele observarse en individuos

jóvenes y puede desarrollarse tanto a partir de una lesión pigmentada previa (lunar o nevus melanocíticas) o a partir de los melanocitos de la piel normal. El pronóstico del melanoma maligno se relaciona directamente con la profundidad histológica del tumor, que suele aumentar a medida que la lesión evoluciona en el tiempo. Por este motivo, resulta fundamental poder establecer un diagnóstico precoz de este tumor [5].

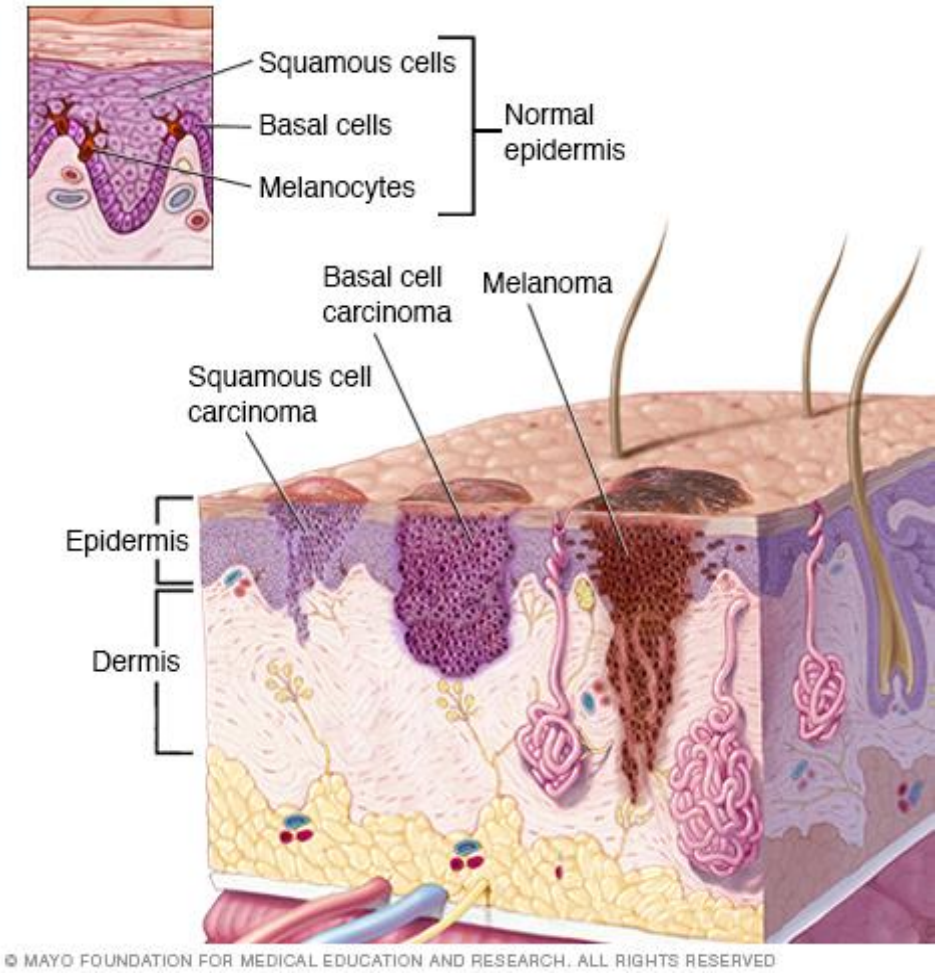


Ilustración 3. Detalles de la epidermis normal y epidermis cancerígeno

En la ilustración 3, se puede observar los detalles de la epidermis normal, y la localización de los melanocitos en su parte inferior. También se puede observar los tres tipos de cánceres de piel que hemos mencionado antes, y el aspecto del tercer tipo (derecha) a saber el melanoma y su profunda extensión en la dermis.

El melanoma maligno es el cáncer de piel más agresivo que existe, tal como se mencionó en la introducción, con una mayor frecuencia de incidencia en mujeres que en hombres. Generalmente aparece en el órgano más grande del cuerpo humano que es la piel. En raras ocasiones, los melanomas aparecen en la boca, en el iris del ojo o en la

retina en la parte posterior del ojo. Se pueden descubrir durante exámenes dentales u oculares. En casos muy excepcionales, un melanoma se desarrolla en la vagina, el esófago, el ano, las vías urinarias y en el intestino delgado [6].

Vamos a definir ahora una de las técnicas más importantes de imágenes que ayuda a diagnosticar las lesiones pigmentadas: la dermatoscopia.

2.2 Dermatoscopia

La Dermatoscopia o ELM (siglas en inglés de Epiluminescence Miscroscopy) es una técnica para la formación de imágenes de la piel no invasiva, que permite la visualización de las características de las neoplasias melanocíticas pigmentadas que no son discernibles por examen con el simple ojo [7].

Ilustración 4. Dermatoscopio convencional

La técnica consiste en el uso de un aparato: el dermatoscopio convencional (ver ilustración 4) que es un estereomicroscopio o microscopio manual pues es un instrumento dotado de un sistema óptico de amplificación de imagen (lentes de aumento) y una fuente de luz convencional o polarizada. Para disminuir la reflexión o refracción de la luz por parte de la epidermis se usa una interfase líquida (agua, aceite o alcohol) lo que permite ver estructuras anatómicas de la epidermis o de la dermis papilar que no son visibles a simple vista [8].

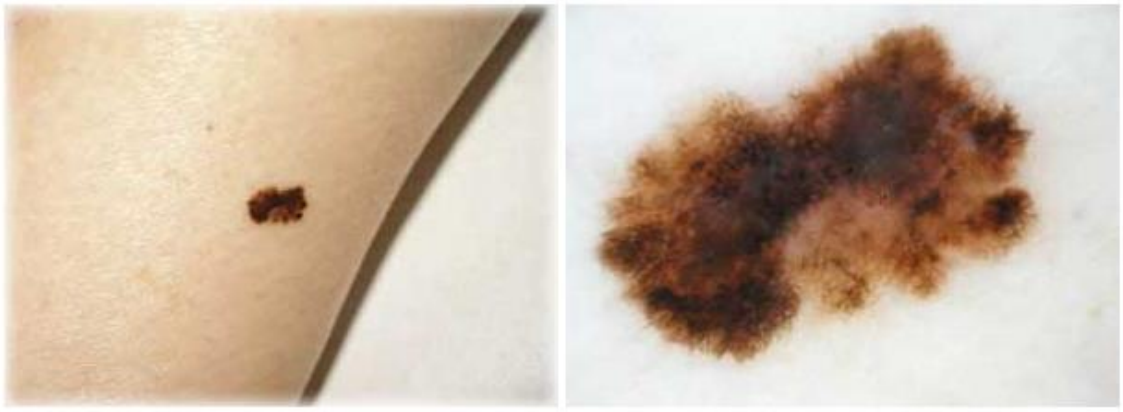


Ilustración 5. Izquierda: Melanoma (imagen clínica). Derecha: Melanoma (Imagen dermatoscópica)

La ilustración 5 muestra la diferencia entre una imagen clínica (tomada con aparato fotográfico normal) y una imagen dermatoscópica. Se puede observar que en la segunda, se ven todos los detalles de la lesión en cuanto a color y textura, facilitando por consiguiente el posterior tratamiento digital de los píxeles de dicha imagen.

Es una técnica de interés creciente para todos los dermatólogos por su bajo coste y la información relevante que permite obtener. Posee una utilidad demostrada en el estudio de los tumores cutáneos, especialmente los pigmentados. Debería utilizarse siempre ante cualquier lesión de esta característica, ya que facilita el diagnóstico diferencial y mejora la precisión diagnóstica del melanoma [9].

En el siguiente apartado, vamos a ver con brevedad los métodos más importantes existentes hoy en día, que nos ofrece la dermatoscopia, en el campo del diagnóstico del melanoma en lesiones pigmentadas.

2.2.1 Métodos existentes de diagnósticos en dermatoscopia

A lo largo la literatura médica, se han propuesto numerosos métodos para la extracción de características a partir de imágenes clínicas de lesiones de la piel. Sin embargo, la extracción de características en imágenes dermatoscópicas es un campo relativamente inexplorado. Los estudios de extracción de características dermatoscópicas en lesiones pigmentadas hasta la fecha incluyen estudios sobre estructuras de pigmentos y glóbulos, y otros estudios sistemáticos sobre los puntos y manchas de la lesión pigmentada [1]. Vamos a ver en los dos próximos puntos, dos de los métodos de diagnóstico importantes que existen en el campo de la dermatoscopia.

2.2.1.1 Regla ABCD de la dermatoscopia

Un estudio importante de análisis ELM es "la regla ABCD de la dermatoscopia". Esta regla se ha basado en la regla ABCD inicial, que es un algoritmo semicuantitativo basado en la evaluación de la forma y la distribución del pigmento, es decir, que evalúa la asimetría en algunos de sus ejes, la regularidad del borde, los colores (si es variado, o no uniforme) y el diámetro de la lesión (no debe ser mayor que 6 mm). Por lo tanto, si detectamos en nuestra piel un lunar o mancha con una o varias de estas características es importante que acudamos rápidamente al dermatólogo, quien podrá realizar un examen más detallado.

En 1994, Stolz et al [10] y Nachbar et al [11] desarrollaron la nueva regla ABCD de la dermatoscopia, basándose en el análisis semicuantitativo de cuatro características extraídas de la superficie microscópica de la piel. Esta regla es parecida a la regla inicial, con la diferencia de que el criterio del Diámetro es reemplazado por las Diferencias estructurales y que para extraer las características se emplea un estereomicroscopio (como se hacía antes) o un dermatoscopio (como se hace en la actualidad).

La regla ABCD de la dermatoscopia, se emplea para diferenciar las lesiones malignas de las benignas. Es una técnica fácil de aprender y rápida de calcular [9], por lo que expertos y principiantes pueden aplicarla, empleando la Ecuación (1):

$$FDS = A \times 1.3 + \mathrm{B} \times 0.1 + C \times 0.5 + D \times 0.5$$

En donde:

• El valor de la **A**simetría, A, depende del número de ejes en los que la lesión es asimétrica. Es decir, trazando dos ejes perpendiculares en la lesión, si la lesión es simétrica, el valor de A será 0, si la lesión es asimétrica respecto a un eje, el valor de A será 1 y si la lesión es asimétrica con respecto a los dos ejes, el valor de A será 2.

• El valor del **B**orde, B, depende del número de segmentos en los que se observa un corte abrupto en el borde de la lesión. Un corte abrupto significa que existe una gran diferencia entre la lesión y la piel en la zona del borde, o dicho de otra manera, que el color de la lesión no se va degradando a medida que se acerca a la piel. Para asignar el valor de B, se debe dividir la lesión en 8 segmentos, siendo cada uno de esos segmentos los que asignen el valor de B y cuando exista un borde abrupto el valor de B aumenta. Por tanto el valor de B varía entre 0 (no hay segmentos con borde abrupto, es decir, que la lesión tiene un borde que se va degradando) y 8 (la lesión tiene un borde abrupto y se diferencia notablemente con respecto a la piel).

• El valor del **C**olor, C, depende del número de colores que intervienen en la lesión. Cuantos más colores, mayor valor de C. Los posibles colores son: blanco, rojo, marrón claro, marrón oscuro, azul grisáceo y negro, por lo que el valor de C varía entre 1 y 6.

• El valor de las **D**iferencias estructurales, D, depende del número de estructuras que posee la lesión. Desde áreas homogéneas, cadenas, venas, puntos y/o glóbulos, por lo que el valor de D varía entre 1 y 5.

• **F**inal **D**ermatoscopy **S**core, FDS, corresponde a la puntuación final de la combinación de las cuatro características que abarca la regla ABCD de la dermatoscopia. Dependiendo del valor de FDS se puede tener: un melanoma maligno, una lesión sospechosa de ser maligna o una lesión benigna.

- ✓ Si FDS excede de 5.45, la lesión tiene una gran probabilidad de ser un cáncer de piel de tipo melanoma maligno. Por tanto, requiere una extirpación lo antes posible y su tratamiento depende del estadio en el que se encuentre el melanoma.

- ✓ Si FDS se encuentra entre 4.75 y 5.45, la lesión tiene probabilidad moderada de ser melanoma maligno. En este caso, es necesaria una vigilancia constante de la evolución de la lesión para verificar si puede convertirse o no en un melanoma maligno.

- ✓ Si FDS es menor de 4.75, es probable que la lesión sea un cáncer de piel de tipo benigno. Por lo que no es necesaria una extirpación, pero quizás requiera un tratamiento diferente, dependiendo del cáncer que sea.

2.2.1.2 New7- Point Checklist

Más tarde, en 1998, Argenziano et al presentó el método New 7 Point Checklist, basado en una simplificación de patrones de análisis ELM presentados en 1989 [12]. Este método es un listado de siete características extraídas de diversas lesiones y que ayudan en la clasificación de las mismas. Las características que forman dicho listado se citan a continuación:

• Retículo pigmentado atípico (Atypical pigment network).

• Zonas de gris (o blanco) - azul (Gray/white - blue areas).

• Patrón vascular atípico (Atypical vascular pattern).

• Transmisión radial (rayas) (Radial streaming (streaks)).

• Pigmentación difusa irregular (manchas) (Irregular diffuse pigmentation (blotches)).

• Puntos y glóbulos irregulares (Irregular dots and globules).

• Patrón de regresión (Regression pattern).

El siguiente paso de este estudio, es el cálculo de un parámetro: Odds Ratio (OR), correspondiente al cociente entre el número de veces que ocurren y el número de veces que no ocurren, las características anteriores en una imágenes dermatoscópicas de lesiones melanocíticas. Puntúa con un valor de 2 a las características que obtengan un resultado de OR superior a 5 y se clasifican como criterios principales y puntúa con un valor de 1 a las características que obtengan un resultado de OR inferior a 5 en cuyo caso se dicen que son criterios menores. Un mínimo 3 como puntuación es requerido para diagnosticar un melanoma es decir un criterio principal más uno menor o bien tres menores. Para entender más, la tabla 1 nos muestra un ejemplo de puntuación.

Características ELM del New7 Point Checklist	Odds ratio (OR)	Puntuación del método
Criterios principales:		
1-Retículo pigmentado atípico	5.19	2
2-Zonas de gris (o blanco) – azul	11.1	2
3-Patrón vascular atípico	7.42	2
Criterios menores:		
4- Transmisión radial (rayas)	3.01	1
5- Pigmentación difusa irregular (manchas)	4.90	1
6- Puntos y glóbulos irregulares	2.93	1
7-Patrón de regresión	3.89	1

Tabla 1. Ejemplo de puntuación método New7 Point Checklist

En el siguiente punto nos fijamos en la segunda característica del método New7-point Checklist. Pues justo este proyecto hace hincapié a esa característica morfológica y que se considera una de las más relevantes identificadas mediante ELM para diagnosticar el melanoma maligno. Es la característica del velo azul-blanco en lesiones pigmentadas.

2.2.2 Velo azul-blanco en lesiones melanocíticas

El velo azul-blanco es una de las características morfológicas más importantes en las lesiones pigmentadas, ya que es el indicador dermatoscópico más significativo del melanoma maligno invasivo, con una sensibilidad de 51% y una especificidad del 97%. Las áreas de este velo se caracterizan por su pigmentación azul confluente, son áreas irregulares, sin estructura, con una película que se recubre con un blanco de "vidrio esmerilado". La ilustración 7 muestra un melanoma con el velo azul-blanco [1].

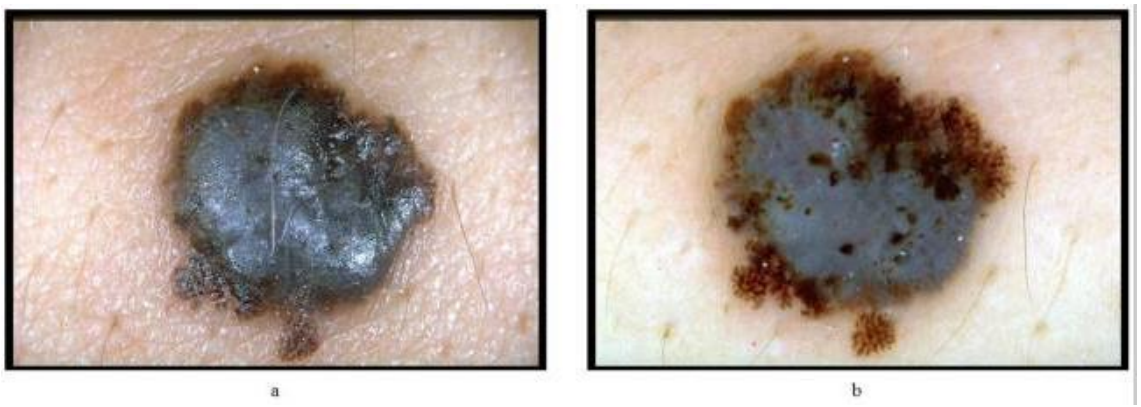


Ilustración 6. Melanoma con velo azul-blanco (a) imagen clínica (b) imagen dermatoscópica

Se puede observar con claridad en la ilustración 6 las regiones de velo azul-blanco descritas tanto en imagen clínica como en otra dermatoscópica (b). A lo largo del desarrollo de este PFC, el procedimiento de la detección de dichas regiones se va a aplicar sobre la imagen dermatoscópica (b).

El hallazgo de estas estructuras blanco-azuladas traduce la presencia de zonas de regresión tumoral en las lesiones y se asocian con una frecuencia muy precisa a melanoma tal como se mencionó antes, aunque también pueden observarse a veces en algunas lesiones benignas con regresión. Sin embargo, la mayoría de los expertos recomiendan extirpar aquellas lesiones en las que las estructuras de velo azul-blanco que ocupen más de un 10% de la superficie.

La importancia de la característica del velo azul-blanco en el diagnóstico del melanoma es muy significativa, ya que según el artículo en el que se basa este proyecto, el porcentaje de las áreas de azul blanco detectadas en una lesión combinada con un simple descriptor de forma dio una sensibilidad del 69,35% y una especificidad del 89,97% en un conjunto de 545 imágenes dermatoscópicas. Por otra parte y teniendo en cuenta la detección del velo azul-blanco como característica principal en el reconocimiento de melanoma en dichas imágenes, la sensibilidad se eleva a 78,20% [1].

De todos los métodos ya existentes en la extracción de características, y en el mejor de nuestro conocimiento, hay un solo estudio sistemático publicado en la detección de velo azul-blanco, y que es el artículo base de este proyecto: *Automatic detection of blue-white veil and related structures in dermoscopy images de M.Emre Celebi, Hitoshi Iyatomi,* ***William V.Stoecker, Randy H.Moss, Harold S.Rabinovitz, Giuseppe Argenziano, H.Peter Soyer***. Por lo que este trabajo engloba dicho estudio, desarrollando un algoritmo que va detectar automáticamente las regiones afectadas con el velo azul-blanco en un conjunto de imágenes dermatoscópicas de lesiones pigmentadas, después de la extracción de unas características de color. En el siguiente capítulo se va a describir con detalle el desarrollo del método empleado.

CAPÍTULO 3: Método

Como ya se dijo en la introducción, en este proyecto presentamos un enfoque de aprendizaje automático para la detección de velo azul-blanco y estructuras relacionadas en imágenes dermatoscópicas. El método implica la clasificación contextual de píxeles utilizando un clasificador de árbol de decisión inducido.

Se va a intentar mejorar los resultados del artículo de base mencionado en la introducción *Automatic detection of blue-white veil and related structures in dermoscopy images de M.Emre Celebi, Hitoshi Iyatomi, William V.Stoecker, Randy H.Moss, Harold S.Rabinovitz, Giuseppe Argenziano, H.Peter Soyer* [1], siguiendo sus pasos y aplicándolos eficientemente.

Este capítulo 3 va a explicar paso a paso las diferentes partes del método aplicado. Está organizado básicamente de la siguiente forma: la sección 3.1 engloba una descripción del conjunto de imágenes dermatoscópicas usadas, así como la fase del pre procesado. En la sección 3.2 se va a ver la parte de extracción de características de color de los píxeles de entrenamiento y prueba obtenidos en el pre procesado. Finalmente en la sección 3.3, obtenemos la clasificación de estos píxeles mediante el algoritmo C4.5.

3.1 Descripción del set de imágenes y pre procesado

3.1.1 Descripción del set de imágenes

El conjunto de imágenes utilizado en este estudio consta de 90 imágenes dermatoscópicas digitales obtenidas del atlas CD-ROM Atlas interactivo de la dermatoscopia [13], que es una colección de imágenes adquiridas en tres instituciones: la Universidad Federico II de Nápoles, Italia, Universidad de Graz, Austria, y la Universidad de Florencia, Italia. Son imágenes en color real con una resolución típica de 768 × 512 píxeles. La distribución de los casos diagnosticados fue como sigue: 60 casos con regiones claras de velo azul-blanco, mayoritariamente son casos de melanoma (ver ilustración 7), 10 casos de Clark nevus sin velo (ver ilustración 8), y 20 casos de Dermal nevus tampoco sin velo azul-blanco (ilustración 9).

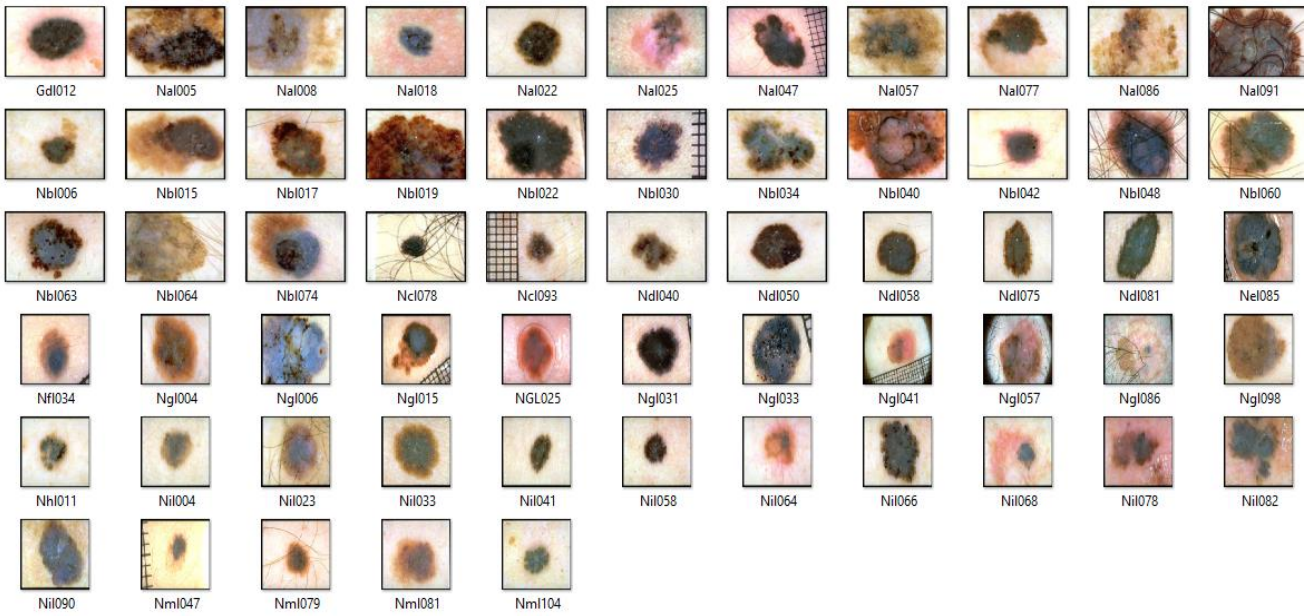


Ilustración 7. Imágenes de melanomas con velo azul-blanco

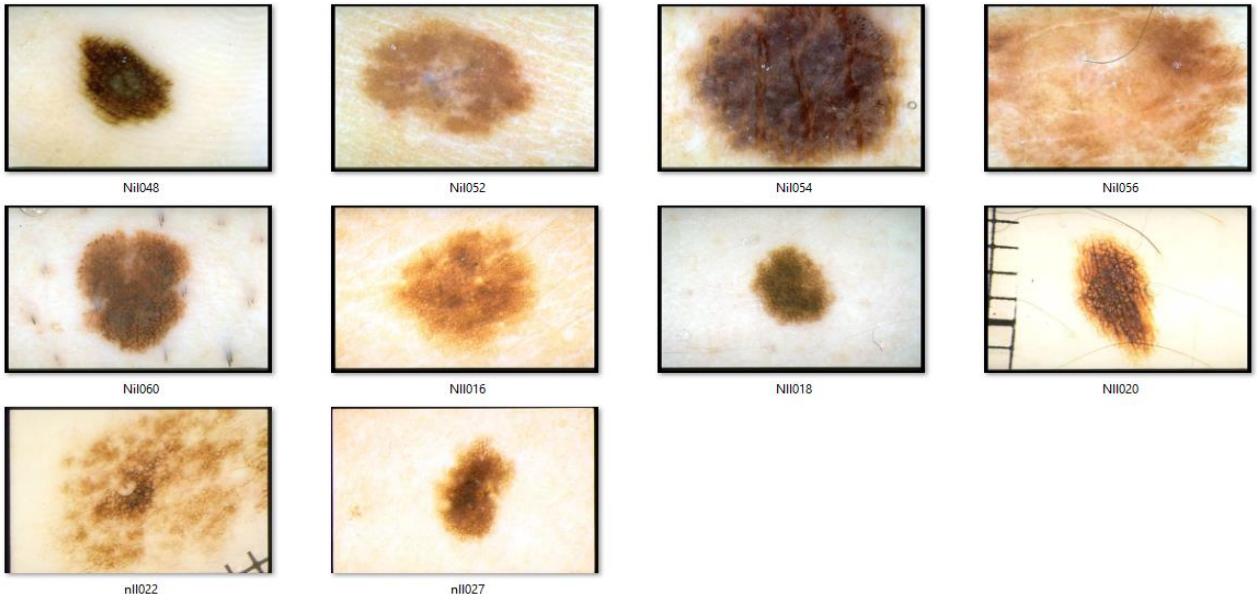


Ilustración 8. Imágenes de Clark nevus sin velo

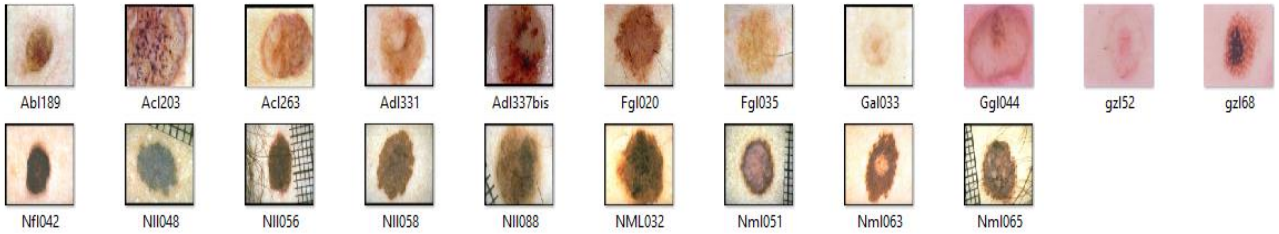


Ilustración 9. Imágenes de Dermal nevus sin velo

Las lesiones se diagnosticaron mediante biopsia de manera histopatológica en casos en los que estaba presente un riesgo significativo de melanoma; de lo contrario, se diagnosticaron mediante examen de seguimiento.

Una vez guardadas dichas imágenes en un directorio donde se les puede acceder desde el programa Matlab para su posterior tratamiento digital, empezamos por la primera fase del desarrollo que es el pre procesado, punto que vemos a continuación en la siguiente sección.

3.1.2 Pre procesado

Esta fase es primordial en cualquier algoritmo de clasificación, ya que antes de la extracción de las características a clasificar, se deben dar algunos pasos previos. En primer lugar, se ejecuta un algoritmo de segmentación que define los bordes de la lesión, separando la piel sana de la zona lesionada. En el caso de nuestro proyecto, una vez identificado el contorno que delimita el borde de la lesión pigmentada de piel, dos pasos más de pre procesado son necesarios, a saber, la determinación del color de la piel del fondo y la selección de los píxeles de entrenamiento y prueba. Estos dos últimos pasos son importantes porque se van a necesitar más adelante en la extracción de las características de color de la lesión (sección 3.2)

Para la determinación del color de "Background" de la piel, se van a determinar dos regiones fuera del contorno de la lesión, la primera tiene un área de 10% del área total de la lesión, y la segunda del 20%. Estas dos regiones se van a omitir, es decir que el color de la piel del fondo deseado se va a determinar a partir del valor promedio de color de los píxeles fuera de dichas regiones omitidas (eso se verá con más detalles en el punto 3.1.2.2). La selección de píxeles de entrenamiento y prueba se hará de forma manual y aleatoria (punto 3.1.2.3).

Todo lo mencionado anteriormente de forma breve se llevará a cabo sobre nuestro conjunto de imágenes y se va a ver por separado en los tres siguientes puntos de manera detallada. La ilustración 10 muestra una visión general del procedimiento de nuestro pre procesado.

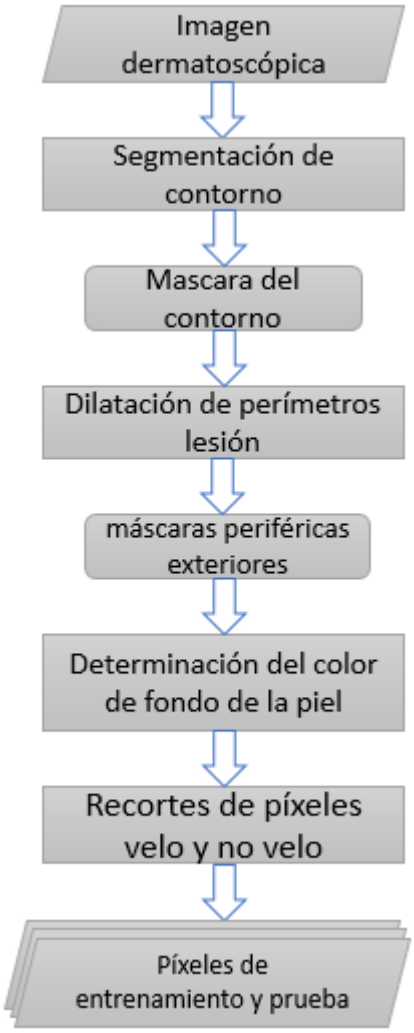


Ilustración 10. Visión general del pre procesado

3.1.2.1 Determinación del contorno de la lesión

El contorno de la lesión se obtuvo de forma automática mediante el uso de umbralización binaria. La imagen segmentada con umbralización binaria es la imagen que diferencia los píxeles pertenecientes a la piel de los pertenecientes a la lesión, haciendo que los valores de la piel correspondan a ceros y que los valores de la lesión correspondan a unos. Para ello se ha realizado la siguiente función de Matlab:

detec_cont.m cuyo diagrama de flujos se define a continuación en la ilustración 11.

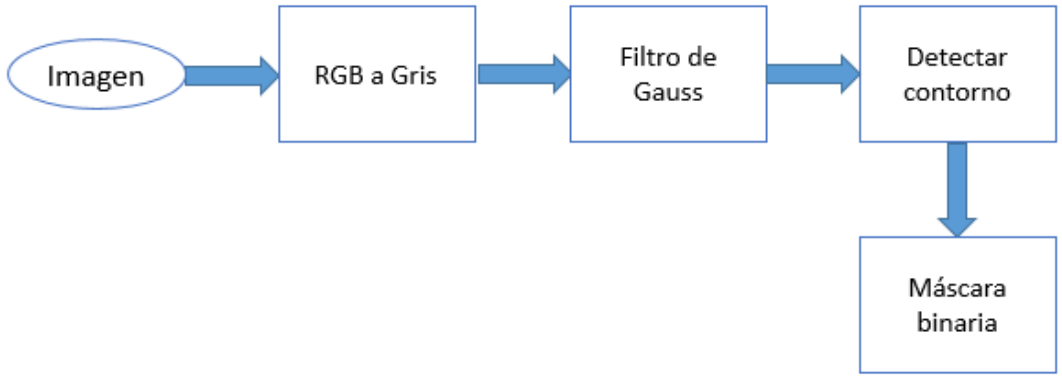


Ilustración 11. Diagrama de flujos de la función **detec_cont.m**

El código Matlab de la función anterior es:

```
function [cont]= detec_cont(Imagen)

%Recorte de las imágenes para eliminar el viñeta negra que viene en el
%atlas
Im_rec= imcrop(Imagen,[34 25 718 466]);
%Pasamos la imagen de escala RGB a escala de grises
Im_gris = rgb2gray (Im_rec);
%3.- Filtrado gaussiano a Im_gris
H = fspecial('gaussian',20,5);

Blurred = imfilter(Im_gris,H,255);
V_medio= mean(mean(Blurred));% obtenemos el valor medio

%obtener región de contorno
cont= Blurred < V_medio;% con el valor medio de la imagen pasamos
% a obtener el contorno de la lesión

end
```

Tabla 2. Código Matlab de la función **detec_cont.m**

A continuación describimos la función:

A la función se le pasa como entrada una de las imágenes dermatoscópicas de nuestro set; variable “Imagen”, y devuelve como salida la máscara binaria del contorno de la lesión de dicha imagen; variable “cont”.
Primero se hace un recorte de la imagen mediante el uso de la función *imcrop*, para quitar la viñeta negra que viene por defecto en todas las imágenes del Atlas, ya que si no se hace este recorte nuestra función va considerar que dicha viñeta forma parte de la lesión, por su color de tono negro. Por eso hemos pasado a la función *imcrop* como parámetros, los valores límites de las coordenadas de la imagen cogiendo el caso

extremo de la viñeta más grande, así nos aseguraremos de hacer un recorte donde se vea solamente la piel más la lesión pigmentada. La ilustración 12 y 13, nos muestra el resultado de esta maniobra, aplicado a un ejemplo de una imagen escogida de las imágenes con velo azul-blanco.

Ilustración 12. Imagen original con viñeta negra en los bordes

Ilustración 13. Imagen con viñeta recortada

Como la imagen recortada está en color RGB, la transformamos a escala de grises (como se puede ver en la ilustración 14) mediante la instrucción *rgb2gray* que convierte las imágenes RGB a escala de grises eliminando la información del tono y de la saturación. El espacio de color RGB (Red, Green, Blue) es la composición del color en términos de la intensidad de los colores primarios de la luz, con este espacio se puede representar un color mediante la mezcla por adición de los tres colores de luz primarios.

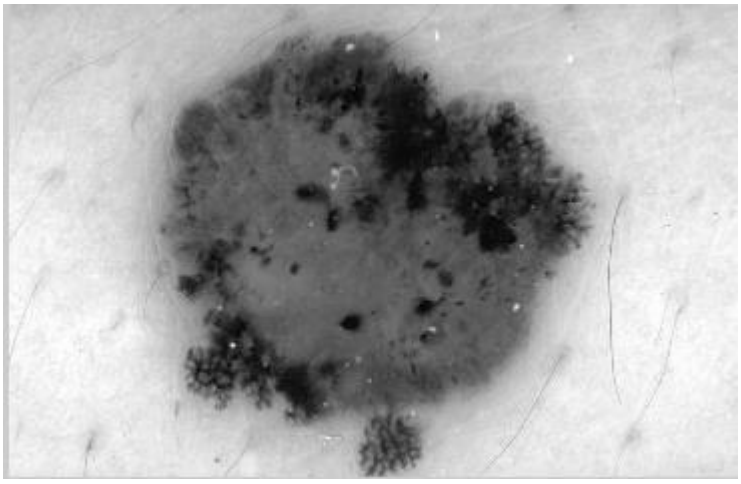


Ilustración 14. Representación de la imagen en escala de grises

Ahora que tenemos la imagen en escala de grises le aplicamos un filtro gaussiano con el fin de suavizar más la imagen. Eso se hace definiendo el argumento del filtro gaussiano con la orden *fspecial*, y después se sigue con la instrucción *imfilter*. Se calcula a continuación el valor promedio de esta imagen suavizada mediante la orden *mean.*

El valor medio de una imagen es simplemente la suma de todos los elementos (valores de píxeles) de esa imagen dividida entre el número total estos elementos, que es el número de filas multiplicado por el número de columnas.

Finalmente, el último paso es obtener el contorno de la lesión de tal forma que se elimine de la imagen aquello que no sea de la propia lesión es decir lo que es piel normal. Para ello se crea una máscara donde el valor de los pixeles será 1 (blanco) en los píxeles lesionados y 0 (negro) donde no hay lesión. Como el color de la lesión es más oscuro que el de la piel esta mascara se puede obtener utilizando una umbralización con umbral el valor promedio de los píxeles de la imagen escala de grises. Ya que en Matlab, si igualamos una matriz a una expresión, el resultado es una matriz donde hay cero donde la expresión es falsa y uno donde es verdadera:

$$Contorno = (Píxel < Valor\ promedio) => Contorno = \begin{cases} 1, & SÍ \\ 0, & NO \end{cases}$$

La ilustración 15 nos muestra un ejemplo de la máscara binaria de la imagen de la ilustración 13.

Ilustración 15. Izquierda la imagen dermatoscópica, derecha aproximación del contorno de la lesión (ejemplo 1)

Vemos otro ejemplo de otra imagen del set, esta vez del conjunto de clark nevus sin velo en la ilustración 16.

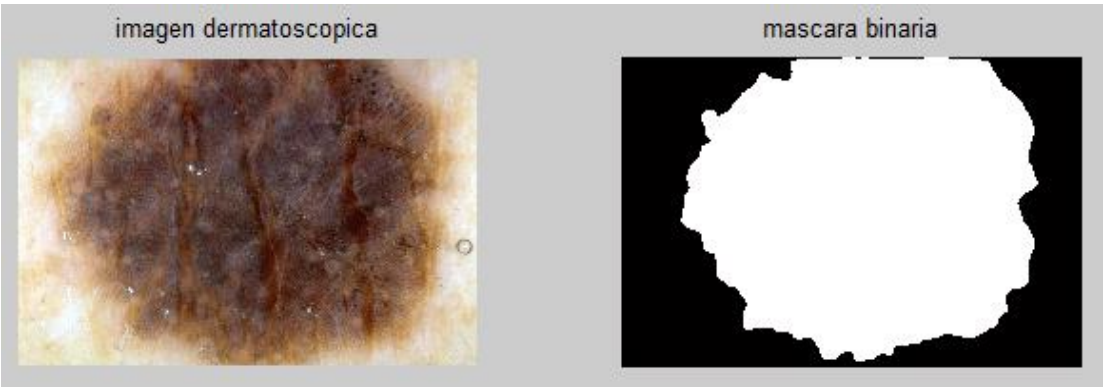


Ilustración 16. Izquierda la imagen dermatoscópica, derecha aproximación del contorno de la lesión (ejemplo 2)

Una vez que tenemos la máscara binaria procedemos a la segunda parte del pre procesado que es la determinación del color de la piel del fondo, lo vemos a continuación.

3.1.2.2 Determinación del color de la piel del fondo

Esta parte es fundamental en el pre procesado, ya que para la extracción de las características de color, el color de la piel del fondo debe ser determinado, pues lo

vamos a utilizar en las ecuaciones de estas características que definiremos en la sección 3.2.

Como ya hemos mencionado, en primer lugar, se omitió la región fuera de la frontera con un área igual a 10% del área de la lesión para reducir los efectos de la inflamación periférica y errores en la determinación de la frontera. El color de la piel del fondo a continuación, se calculó como el promedio de color sobre la próxima región fuera de la frontera con otra área igual al 20% del área de la lesión. Tampoco se incluyeron en el cálculo aquellos píxeles que no pertenecen a la piel (cuadros de imagen negro, reglas, pelos, y burbujas). La ilustración 17 y 18 nos muestran ejemplos de eso píxeles a omitir.

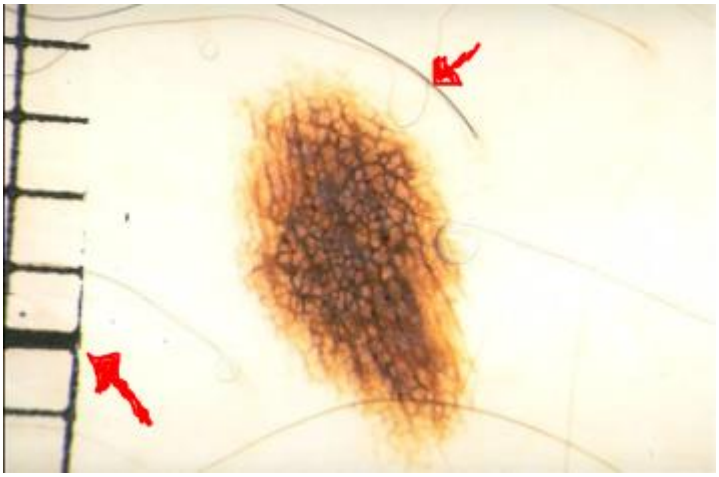


Ilustración 17. Ejemplo cuadros negros de imagen y pelos a omitir en el cálculo del color de la piel

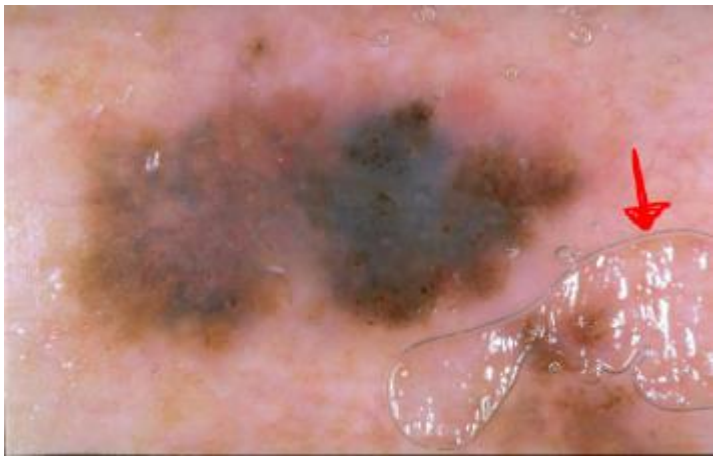


Ilustración 18. Ejemplo de burbujas a omitir en el cálculo del color de piel

Para este fin forzamos que estos píxeles omitidos fueron aquellos que no satisfacen la siguiente regla empírica [14]:

$$(R> 90 \cap R> B \cap R> G);$$

Donde R, G y B denotan los valores de rojo, verde y azul, respectivamente, del pixel en cuestión. En cuando a los 10% y 20% de áreas fuera de la lesión se determinaron a partir de la máscara binaria usando la propiedad morfológica de dilatación tal como vamos a ver más adelante en el código de la función Matlab empleado para este paso del pre procesado.

Vamos a denotar el valor promedio del color de la piel del fondo (índice S de skin en inglés) por (R_S, B_S, y G_S) donde R_S es el valor rojo promedio de los píxeles de la piel, B_S es el valor azul promedio de los píxeles de la piel, y G_S es el valor verde promedio de los píxeles de la piel.

detec_color_BG.m es el nombre de nuestra función Matlab empleada cuyo diagrama de flujos resumido se define a continuación en la ilustración 19:

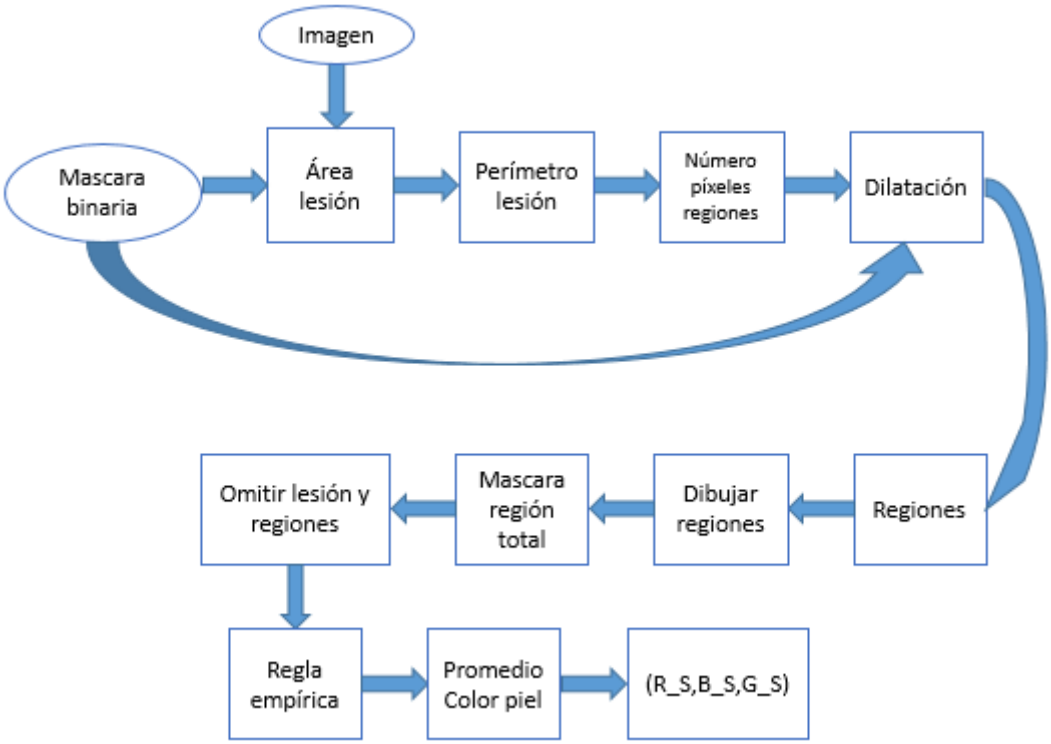


Ilustración 19. Diagrama de flujos de la función **detec_color_BG.m**

El código Matlab de la función anterior es:

```
function [R_S,B_S,G_S]= detec_color_BG(bin_imagen,Imagen)

%1)-->obtenemos las máscaras binarias de las dos regiones


%medimos el área de la lesión
Area_les= nnz(bin_imagen);
%medimos el perímetro de la lesión
g=bwperim(bin_imagen);
%primer perímetro
Perim1= nnz(g);
%calculamos el número de pixeles en ambas regiones Reg1: 10% del área
y
%Reg2: 20% del área:
npixel1=0.1*(Area_les/Perim1);
npixel2=0.2*(Area_les/Perim1);
%creamos los elementos estructurales para las regiones 1 y 2:
SE1= strel('disk',round(npixel1));
SE2= strel('disk',round(npixel2));
%dilatamos la máscara binaria el 10% de los pixeles del área:
dil1=imdilate(bin_imagen,SE1);
%dilatamos un 20% más las imagen ya dilatada anteriormente:
dil2=imdilate(dil1,SE2);
%obtenemos las dos regiones deseadas:
Reg1=xor(dil1,bin_imagen);
Reg2=xor(dil2,dil1);
%Dibujamos las regiones obtenidas juntos a la mascará de la lesión
subplot(1,3,1),imshow(bin_imagen),title('Región lesión');
subplot(1,3,2),imshow(Reg1),title('Región 1(10%)');
subplot(1,3,3),imshow(Reg2),title('Región 2(20%)');



%2)-->dibujamos las dos regiones omitidas sobre la imagen original
%Obtenemos los vectores de coordenadas de píxeles de las regiones a
dibujar
[row1,col1]=find(Reg1);
[row2,col2]=find(Reg2);
%calculamos la longitud de los vectores de coordenadas de los píxeles
l1=length(row1);
l2=length(row2);
%Recorte de las imágenes para eliminar el viñeta negra que viene en el
atlas
im= imcrop(Imagen,[34 25 718 466]);
%Recorremos los píxeles a colorear y le damos color gris para región
10% y
%color blanco para la región del 20%
for j=1:l1
    im(row1(j),col1(j),1)=128;
    im(row1(j),col1(j),2)=128;
    im(row1(j),col1(j),3)=128;
end
for i=1:l2
    im(row2(i),col2(i),1)=255;
    im(row2(i),col2(i),2)=255;
    im(row2(i),col2(i),3)=255;
```

```
end
figure,imshow(im);



%3)-->obtenemos y dibujamos la imagen omitiendo la lesión y el 10% y
20% de ella
%obtenemos la región total omitida (10%+20%+área lesión):
Regtot=dil2;
%convertimos la imagen para dar un valor 0 a las regiones a omitir:
Reg=not(Regtot);
%Recorte de las imágenes para eliminar el viñeta negra que viene en el
atlas
Im_rec= imcrop(Imagen,[34 25 718 466]);
%calculamos los valores R, G, y B de la imagen
Im_R=Im_rec(:,:,1);
Im_G=Im_rec(:,:,2);
Im_B=Im_rec(:,:,3);
%multiplicamos dichos valores por la máscara creada para incluir la
región omitida
R=uint8(Reg).*Im_R;
G=uint8(Reg).*Im_G;
B=uint8(Reg).*Im_B;
%Volvemos a obtener la imagen en RGB, con la región total omitida
(area%lesion+10%+20%)
I=cat(3,R,G,B);
%dibujamos la imagen RGB obtenida
figure,imshow(I);



%4)-->Vamos ahora a proceder al cálculo del color de fondo de la piel
(R_S,B_S,G_S)
%obtenemos la máscara de la regla empírica para los píxeles a omitir
E1=(R>90);
E2=(R>B);
E3=(R>G);
E=E1.*E2.*E3;
%calculamos el número total de píxeles a tener en cuenta
npix=sum(sum(E));
%Obtenemos los valores medios de color (R_S,B_S,G_S) de la piel fuera
de la lesión
R_S=sum(sum(double(R).*E))/npix;
B_S=sum(sum(double(B).*E))/npix;
G_S=sum(sum(double(G).*E))/npix;




end
```

Tabla 3. Código Matlab de la función **detec_color_BG.m**

A continuación describimos la función:

La función **detec_color_BG.m** tiene como parámetros de entrada la imagen dermatoscópica “Imagen”, y también la máscara binaria de dicha imagen “bin_imagen”,

obtenida en el paso anterior del pre procesado mediante la función **detec_cont.m**, que ya hemos descrito. Como salida nos devuelve los valores promedios de color de la piel del fondo requeridos (R_S, B_S, G_S).

La función se ha dividido en 4 bloques dependientes entre sí, aunque cada uno realiza una cosa distinta. El primer bloque tiene como objetivo obtener las máscaras binarias de las regiones 1 y 2 del 10% y 20% del área total de la lesión respectivamente. Estas regiones son las denominadas variables auxiliares "Reg1" y "Reg2" en el código. Para ello lo primero que hace la función es calcular el área y el perímetro de la máscara binaria. Mediante las instrucciones *nnz* y la instrucción *bwperim*. La razón por la cual se han calculado el área y el perímetro es para aplicar esa ecuación obvia:

$$Ancho\ del\ segmento \times Perímetro\ lesión\ = Área\ lesión$$

A nosotros como nos interesa calcular anchura de segmento para el 10% y el 20 % del área, calculamos esas dos cantidades:

$$Anchura\ segmento\ región1\ =\ 0.1 \times (\frac{\text{Área}}{\text{Perímetro}})$$

$$Anchura\ segmento\ región2\ =\ 0.2 \times (\frac{\text{Área}}{\text{Perímetro}})$$

Con estos valores se obtienen los elementos estructurales "SE1" para la región 1 y "SE2" para la región 2, necesarios para la posterior dilatación de la máscara binaria, eso se hace mediante la orden *strel.* Notar que le hemos pasado a *strel* como primer parámetro la cadena 'disk', ya que nos interesa que la dilatación sigue la forma circular de la región de la lesión. Una vez obtenidos los "SE1", "SE2", dilatamos el contorno de la lesión. Las regiones deseadas se calculan después mediante una simple operación lógica XOR:

- La región 1 es el resultado XOR entre la máscara de la lesión y su dilatación 10%.
- La región 2 es el resultado XOR entre la dilatación de la mascará 10% y la dilatación de la misma 20%.

Recordar que la dilatación es una operación matemática morfológica que es una teoría y técnica para el análisis y tratamiento de las estructuras geométricas, basada en la teoría de conjuntos donde siendo E un espacio euclidiano o una cuadrícula entera y A una imagen binaria en E, la dilatación de A por el elemento B se define por [15]:

$$A \oplus B = \{z \in E | B_z^s \cap A\ \neq\ \emptyset\}$$

Donde B_z^s denota la simetría de B

$$B^s = \{x \in B | -x \in B\}$$

El bloque 1 de la función termina por dibujarnos estas regiones para el caso de una máscara binaria de una de las lesiones, juntos con dicha máscara, La ilustración 20 nos lo muestra.

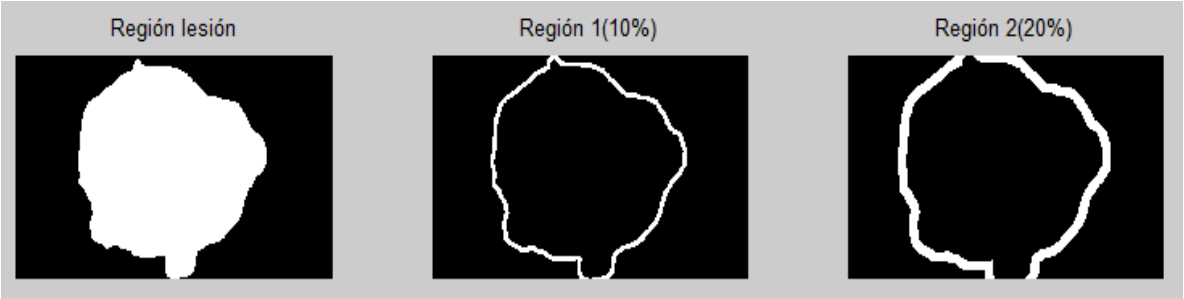


Ilustración 20. De derecha a izquierda: máscara lesión, región 1, región 2

El bloque 2 de la función **detec_color_BG.m** se encarga de dibujarnos esas regiones coloreadas (región1 en gris y región 2 en blanco) sobre la imagen dermatoscópica de entrada, con el fin de visualizar donde consideramos los píxeles a tener en cuenta para el cálculo del color promedio de piel de fondo, ver la ilustración 21.

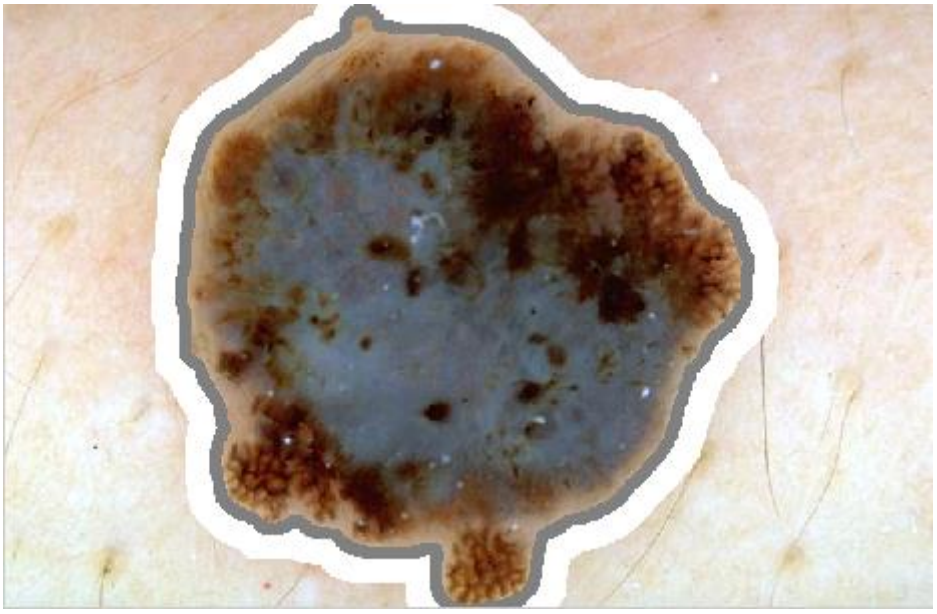


Ilustración 21. 10% (gris) y 20% (blanco) regiones fuera de la lesión

Para ello lo que vemos en el bloque 2 es básicamente buscar los píxeles que valen uno en las dos regiones mediante la orden *find* y guardar sus correspondientes filas y columnas en dos variables "row" y "col", para después recorrer la imagen original, y dar los valores 128 (gris) y 255 (blanco) de R, G, B en estas posiciones guardadas.

El tercer bloque de nuestra función nos omite la región total, que es igual a la región de la lesión más la región 10% y la región 20%. Es fundamental para el posterior cálculo del valor promedio de color de la piel que se va a ver en el bloque 4. Primero se puede ver que la región total a omitir es justamente la variable "dil2", ya que "dil2" es en efecto la dilatación total que incluye todo lo que hay que omitir. Por lo que la negamos mediante la orden *not*, para obtener la máscara deseada, y multiplicamos esta última por los valores R, G, B de la imagen, así solo se podría tratar los píxeles fuera de esa región total. Para poder visualizar la región total omitida, concatenamos los tres valores RGB, y hacemos un *imshow* (ilustración 22).

Ilustración 22. Región total omitida en negro (valor lógico 0)

El último bloque 4 de la función es el más importante ya que es el que nos determina los (R_S, B_S, G_S) que es el valor promedio del color de la piel del fondo fuera de la lesión. Pero primero, no hay que olvidar de implementar la regla empírica que hemos mencionado antes para poder omitir los píxeles que no tienen que ver con la piel. Eso lo hicimos mediante la intersección de las tres expresiones lógicas (R>90), (R>B), (R>G). En Matlab es lo mismo que asignar estas expresiones a matrices lógicas y luego hacer un producto matricial entre los tres, operación que se realizó y guardó en la variable "E" donde se ha guardado la matriz lógica deseada.

Por último, para calcular el valor promedio de color, hemos hecho uso de la expresión clásica en operaciones matriciales en Matlab: *sum (sum ( ))*, como se puede ver en las

últimas líneas del código de la función. Lo que hace básicamente la instrucción anterior es devolvernos la suma de todos los valores de los píxeles de la matriz (o imagen). El valor promedio de cada valor de rojo, azul y verde es entonces la suma de los valores de cada matriz R, G y B dividido por el número total de píxeles a tener en cuenta. Dicha operación se aplica sobre las variables *R, G, B,* donde se han guardado las imágenes con la máscara total de la región a omitir ya incluida. Los guardamos en R_S, B_S y G_S respectivamente y los devolvemos como salida de la función.

Ahora que tenemos el color de la piel del fondo, nos queda un solo paso para acabar la fase del pre procesado, que es la selección de los píxeles de entrenamiento y de prueba, para extraer de ellos las características que vamos a entrenar y clasificar en la sección 3.3. Pero ahora vemos la selección de esos píxeles en el siguiente punto.

3.1.2.3 Píxeles de entrenamiento y de prueba

Con el fin de seleccionar los píxeles de entrenamiento y prueba para la clasificación de sus características hemos seleccionado todas las imágenes que disponemos en nuestro set. Así tendremos mejor entrenamiento para una clasificación más precisa y óptima.

Como ya hemos mencionado en la descripción de nuestro set de imágenes dermatoscópicas (punto 3.1.1), tenemos 90 imágenes en total; sesenta de estas imágenes tienen regiones de puro velo de tamaño significativo, pero también regiones de no velo de tamaño considerable. Por otra parte disponemos también de otras 10 imágenes de Clark nevus sin velo, y otros 20 de Dermal nevus también sin velo. En cada una de esas imágenes, se determinaron manualmente pequeñas regiones que contienen ya sea píxeles de velo o no velo. Los píxeles de entrenamiento y prueba fueron seleccionados al azar de estas regiones determinadas de forma manual. El método de selección fue diseñado para garantizar una distribución equilibrada de las dos clases (velo y no velo) en el conjunto de entrenamiento [16].

Por consiguiente vamos a tener en total 150 recortes de píxeles distribuidos como sigue:

- De los 60 imágenes (Set1) que contienen velo azul-blanco tendremos 120 recortes, ya que en cada una de esas imágenes hacemos dos recortes diferentes, uno en la región de puro velo, y otro en la región de puro no-velo

- De los 10 imágenes de Clark (Set2) nevus sin velo, hacemos 10 recortes uno en cada imagen, y obviamente serán píxeles de no velo

- De los 20 imágenes de Dermal nevus (Set3), otros 20 recortes de píxeles que clasificamos como no velo.

Resumiendo, nuestros recortes están organizados como viene en la tabla 4:

	Set1(Blue-White)	Set2(Clark)	Set3(Dermal)
Velo	60	0	0
No velo	60	10	20

Tabla 4. Distribución de los recortes de velo y no velo

Es decir un total de 60 recorte de velo, y 90 de no velo.

A la hora de seleccionar los píxeles de velo, cabe notar que para cada lesión, se han tenido en cuenta dos características diferentes, velo azul-blanco primario y estructuras relacionadas con velo, características estudiadas en el campo de la dermatología (WVS). Se dice que un velo tiene una característica primaria si el velo es el rasgo más característico de melanoma con color azul blanquecino claro, es decir, la característica presente en la lesión es más reconocible y específica para melanoma. Los velos de color gris o azul-gris o cualquier velo que carecían de la película blanquecina visto en el velo clásico, fueron identificados como estructuras relacionadas con velo. Estos también se consideraron en este estudio como velo.

A continuación explicamos cómo se ha procedido a la selección de esos recortes:

Primero nos situamos en el directorio de Matlab donde tenemos nuestro conjunto de imágenes dermatoscópicas y las leemos una a una mediante el comando *imread*, guardándolas en variables. Después ejecutamos el comando *imcrop* de variable de la imagen. Se nos abre la ventana de la imagen, y mediante el cursor seleccionamos manualmente las regiones sea de velo o no velo, y guardamos los recortes en variables auxiliares. Para entenderlo mejor vemos un ejemplo de secuencia descrita:

```
>> im=imread('Nbl063.jpg');

>> Velo=imcrop(im);
```

Después del segundo comando se nos abre la ventana (ilustración 23).

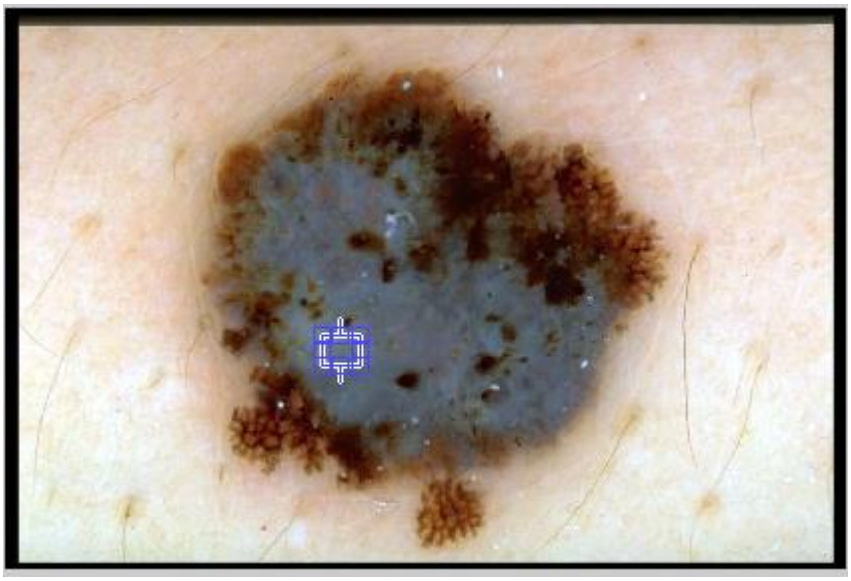


Ilustración 23. Ventana de la imagen con el recorte de velo seleccionado

Vemos en la ilustración 23 el recorte seleccionado en una clara región de puro velo. Lo que sigue después es pulsar sobre el botón derecho del ratón sobre el cuadro y elegir la opción "crop image". Esto hace que se nos guarda el recorte en la variable im. La imagen im del recorte anterior se muestra en la ilustración 24.

Ilustración 24. Imagen del recorte de la ilustración 23

El mismo procedimiento se repite para todos los demás recortes. Al final obtuvimos 150 recortes que hemos guardado en variables y guardado en ficheros data de Matlab (.mat) para su posterior uso en la extracción de características de la sección 3.3. La ilustración 25 ilustro el "workspace" de esos recortes distribuidos tal como hemos descrito antes:

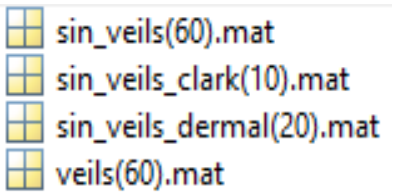


Ilustración 25. Total de los recortes guardados en fichero de datos Matlab

Después de la selección de los píxeles de entrenamiento y prueba, falta extraer las características de estos píxeles que se entrenarán y probarán para la clasificación. Con eso damos por terminado la fase del pre procesado. Vemos la extracción de las características de color en la siguiente sección.

3.2 Extracción de características

En esta sección vamos a definir las características de color que vamos a aplicar tanto a los píxeles de entrenamiento y prueba que hemos seleccionado manualmente en el pre procesado. También las vamos a aplicar en la última parte del trabajo sobre nuestro conjunto de imágenes enteras para detectar las regiones de velo azul-blanco, una vez obtenidas las reglas de decisión

En este estudio vamos a ver un total de 15 características de color para caracterizar los píxeles de la imagen, que vamos a repartir en dos grupos. El primero engloba solo 3 de estas características, que son las características de color absoluto, los vemos en el punto 3.2.1. El segundo grupo de características contiene el resto de características que son 12 características de color relativo, estas las definiremos en el punto 3.2.2.

Como denotación, las 15 características (Features en inglés) de este estudio las llamaremos de F1 a F15. La tabla 5 resume todas las características.

Características de color	Descripción
$F_1 = \frac{R_L}{R_L + G_L + B_L}; \quad F_2 = \frac{G_L}{R_L + G_L + B_L}; \quad F_3 = \frac{B_L}{R_L + G_L + B_L}$	Coordenadas de cromaticidad
$F_4 = \frac{R_L}{R_S}; \quad F_5 = \frac{G_L}{G_S}; \quad F_6 = \frac{B_L}{B_S}$	Relación R, G, B relativa
$F_7 = \frac{F_4}{F_4 + F_5 + F_6}; \quad F_8 = \frac{F_5}{F_4 + F_5 + F_6}; \quad F_9 = \frac{F_6}{F_4 + F_5 + F_6}$	Relación R, G, B relativa normalizada
$F_{10} = R_L - R_S; \quad F_{11} = G_L - G_S; \quad F_{12} = B_L - B_S$	Diferencia R, G, B relativa
$F_{13} = \frac{F_{10}}{F_{10} + F_{11} + F_{12}}; \quad F_{14} = \frac{F_{11}}{F_{10} + F_{11} + F_{12}}; \quad F_{15} = \frac{F_{12}}{F_{10} + F_{11} + F_{12}}$	Diferencia R, G, B relativa normalizada

Tabla 5. Definición de las características de color

Donde (R_L, B_L, G_L) son los valores RGB de los píxeles de la lesión y (R_S, B_S, G_S) es el promedio del color de la piel del fondo en el espacio RGB calculado en el pre procesado.

3.2.1 Características de color absoluto

El color absoluto de un pixel se cuantificó por sus coordenadas de cromaticidad F1, F2, y F3 (ver tabla 5). Una ventaja de F1, F2, y F3 sobre los valores brutos de R, G y B es que mientras que los primeros son invariantes la intensidad de iluminación [17], los últimos no lo son. Esta invariancia es esencial para hacer frente a las imágenes que se adquieren en condiciones no controladas de su formación.

3.2.2 Características de color relativo

El Color relativo se refiere al color de un píxel de la lesión en comparación con el color medio de la piel del fondo. Un total de 12 características de color relativos se extrajeron de cada píxel (ver tabla 5). Las características de color relativos ofrecen varias ventajas:

- En primer lugar, se compensan las variaciones en las imágenes causadas por iluminación y / o digitalización.

- En segundo lugar, igualan las variaciones de color de la piel normal entre los individuos.

- En tercer lugar, el color relativo es más natural desde el punto de vista perceptual. Estudios recientes [19] han confirmado la utilidad de las características de color relativos en el tratamiento y análisis de imágenes de lesión de piel.

Una vez definidas las características de color vamos a ver cómo hemos implementado en Matlab la aplicación de estas características sobre los píxeles que hemos recortado antes, para después en la sección 3 entrenarlas y clasificarlas. Esa implementación se verá a continuación en el siguiente 3.2.3.

3.2.3 Implementación en Matlab

Para la extracción de las características de color definidas anteriormente de los recortes de píxeles de entrenamiento y prueba, se ha definido una función nueva en Matlab. La hemos llamado **features_color_croped.m** cuyo diagrama de flujos resumido se define a continuación en la ilustración 26.

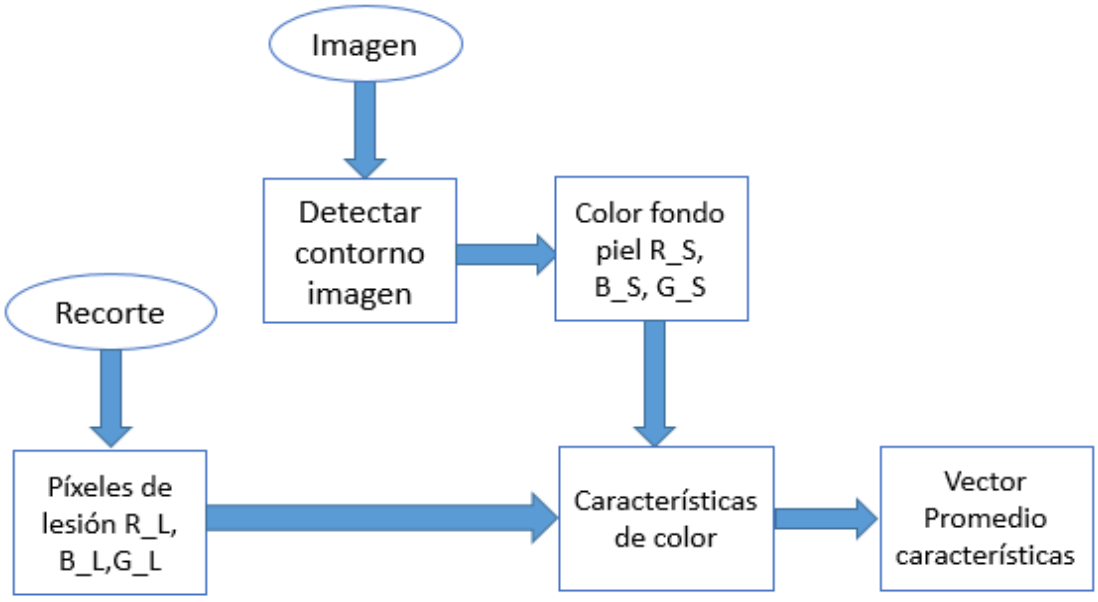


Ilustración 26. Diagrama de flujos de la función **features_color_croped.m**

El código Matlab de la función anterior es:

```
function [MF]= features_color_croped(Imagen,Rec)


%1)-->Obtención de los valores RGB tanto de los píxeles de la lesión
como
%del promedio color de piel de fondo
%Obtenemos la máscara binaria de la lesión
g=detec_cont(Imagen);
%calculamos las matrices R,G,B de la imagen
R_L=Rec(:,:,1);
G_L=Rec(:,:,2);
B_L=Rec(:,:,3);
%obtenemos el valor medio de color de piel del fondo
[R_S,B_S,G_S]= detec_color_BG(g,Imagen);


%2)-->Obtención de los promedios de características de color

%Coordenadas de cromaticidad
F1=R_L./(R_L+G_L+B_L);
[N M]=size(F1);%obtenemos el número de filas y columnas de la matriz F1
MF1=sum(sum(F1))/(N*M);%calculamos el valor promedio de todos los
valores de F1
F2=G_L./(R_L+G_L+B_L);
[N M]=size(F2);
MF2=sum(sum(F2))/(N*M);
F3=B_L./(R_L+G_L+B_L);
[N M]=size(F3);
MF3=sum(sum(F3))/(N*M);
%relacion R,G,B relativa
F4=R_L/R_S;
[N M]=size(F4);
MF4=sum(sum(F4))/(N*M);
F5=G_L/G_S;
[N M]=size(F5);
MF5=sum(sum(F5))/(N*M);
F6=B_L/B_S;
[N M]=size(F6);
MF6=sum(sum(F6))/(N*M);
%relacion R,G,B relativa normalizada
F7=F4./(F4+F5+F6);
[N M]=size(F7);
MF7=sum(sum(F7))/(N*M);
F8=F5./(F4+F5+F6);
[N M]=size(F8);
MF8=sum(sum(F8))/(N*M);
```

```
F9=F6./(F4+F5+F6);
[N M]=size(F9);
MF9=sum(sum(F9))/(N*M);
%diferencia R,G,B relativa
F10=double(R_L)-R_S;
[N M]=size(F10);
MF10=sum(sum(F10))/(N*M);
F11=double(G_L)-G_S;
[N M]=size(F11);
MF11=sum(sum(F11))/(N*M);
F12=double(B_L)-B_S;
[N M]=size(F12);
MF12=sum(sum(F12))/(N*M);
%diferencia R,G,B relativa normalizada
F13=F10./(F10+F11+F12);
[N M]=size(F13);
MF13=sum(sum(F13))/(N*M);
F14=F11./(F10+F11+F12);
[N M]=size(F14);
MF14=sum(sum(F14))/(N*M);
F15=F12./(F10+F11+F12);
[N M]=size(F15);
MF15=sum(sum(F15))/(N*M);

MF=[MF1,MF2,MF3,MF4,MF5,MF6,MF7,MF8,MF9,MF10,MF11,MF12,MF13,MF14,MF15];
Endx
```

Tabla 6. Código Matlab de la función **features_color_croped.m**

A continuación describimos la función:

La función **features_color_croped.m** tiene dos parámetros de entrada y un parámetro de salida. Las entradas son el recorte de la lesión, al que se le va a extraer las características de color (variable "Rec") y la imagen entera (variable "Imagen") que la función va a utilizar para extraer de ella el color de la piel sana ósea el color de fondo de la piel. Esta imagen es la misma que contiene la lesión que se le ha realizado el recorte de la variable "Rec" de la muestra de píxeles de entrenamiento, sea de velo o de no velo. Como salida nos devuelve un vector línea; variable "MF" de 15 elementos correspondientes a los valores promedios de las 15 matrices de características de color que hemos definido.

La función tiene dos partes. En la primera parte se obtienen las matrices R_L, B_L y G_L, que son los valores del espacio RGB de los píxeles de la lesión, y que vamos a necesitar para calcular las matrices de características de color. Como estamos en fase de entrenamiento, estamos usando los recortes es decir pequeñas regiones de velo o no velo, calculamos directamente los valores R_L, G_L, B_L sin falta de ninguna máscara, ya que estamos dentro de la lesión. En esta parte también se hace una llamada a la función **detec_color_BG.m** para obtener el promedio color de la piel del fondo (R_S, B_S, G_S), necesario también para el cálculo de características de color. De aquí la

razón de haber pasado la imagen dermatoscópica entera como entrada a la función, que lo necesita la función **detec_color_BG.m.**

La segunda parte de la función calcula los valores promedios de cada una de las 15 características de color (tabla 5). La razón por la cual estamos calculando el promedio de los valores de las características y no calcularlas tal cual, es para simplificar el estudio, porque cada una de las características es realmente una matriz de valores calculados de los píxeles del recorte en cuestión. No obstante, necesitamos tener un solo valor promedio por característica, para nuestro set de entrenamiento.

Lo que nos conlleva a que justo después de hacer el cálculo de las características, calculamos su promedio usando la orden sum (sum ()) (ver tabla 6 del código). Cabe notar también que a la hora de definir las características usamos el operador ". /", puesto que estamos haciendo divisiones matriciales.

Para cada recorte entonces tendremos su vector de las características de color [MF1,…, MF15]. Como ya sabemos de la sección de píxeles de entrenamiento y prueba que disponemos de 150 recortes (60 de velo y 90 de no velo) eso es que hemos obtenido 150 vector de características. Esos vectores son los que vamos a entrenar con nuestro algoritmo de clasificación para obtener un clasificador o árbol de decisión. Así nos introducimos a la última sección de este capítulo, que vemos a continuación.

3.3 Clasificación de los píxeles

Para la clasificación de los píxeles tuvimos varias opciones a la hora de elegir el algoritmo a implementar. Como clasificadores populares utilizados en tareas de clasificación de píxeles que incluyen los k-vecinos más cercanos, clasificador bayesiano, las redes neuronales artificiales, o las máquinas de vectores de soporte.

En este estudio, se utilizó un clasificador de árbol de decisión para clasificar los píxeles de la imagen en 2 clases: velo y no velo. Según el artículo en que se ha basado este trabajo, la motivación de esta elección era doble. En primer lugar, los clasificadores de árboles de decisión generan reglas fáciles de entender, lo cual es importante para la aceptación clínica de un sistema de diagnóstico asistido por ordenador. En segundo lugar, son rápidos para entrenar y aplicar. Se ha optado entonces por usar el muy conocido **algoritmo C4.5** para inducir el árbol de decisión. Lo vemos en el siguiente apartado.

3.3.1 Algoritmo C4.5

Dado un conjunto de entrenamiento grande, los clasificadores de árboles de decisión, en general, generan reglas de decisión complejas, y aunque funcionan bien en los datos de entrenamiento, no generalizan bien los datos invisibles [20]. En tales casos, se dice que el modelo clasificador ha sobre ajustado los datos de entrenamiento. El algoritmo C4.5 impide este sobreajuste mediante la poda del árbol inicial, es decir, mediante la identificación de los subárboles que contribuyen poco a la exactitud de predicción, por lo que sustituye cada uno de eso subárboles por una hoja [21].

El algoritmo C4.5 se entrenó entonces con las características de color de los píxeles, que hemos seleccionado manualmente (Sección 3.2). Pero antes de ver el árbol de decisión obtenido o cualquier resultado, vemos con detalle los pasos que hemos seguido, durante nuestro desarrollo de esta parte. En el siguiente sub apartado vemos qué es el algoritmo C4.5 y cómo funciona. Luego en 3.3.1.2, presentamos su implementación en Matlab para nuestro proyecto.

3.3.1.1 Teoría y Descripción

El algoritmo C4.5 pertenece a la familia TDIDT, de los "Top Down Induction Trees", que son métodos inductivos del Aprendizaje Automático, pues aprenden a partir de ejemplos preclasificados. En Minería de Datos, se utiliza para modelar las clasificaciones en los datos mediante árboles de decisión [22].

3.3.1.1.1 Construcción de los árboles de decisión

Los árboles TDIDT, a los cuales pertenecen los generados por el ID3 y por el C4.5, se construyen a partir del método de Hunt. El esqueleto de este método para construir un árbol de decisión a partir de un conjunto T de datos de entrenamiento es muy simple. Sean las clases {C1, C2,. . ., Ck}. Existen dos posibilidades:

- T contiene uno o más casos, todos pertenecientes a una única clase Cj: El árbol de decisión para T es una hoja identificando la clase Cj.

- T contiene casos pertenecientes a varias clases: En este caso, la idea es refinar T en subconjuntos de casos que tiendan, o parezcan tender hacia una colección de casos pertenecientes a una única clase.

3.3.1.1.2 Origen algoritmo C4.5

El algoritmo c4.5 fue desarrollado por JR Quinlan en 1993, como una extensión (mejora) del algoritmo ID3 que desarrollo en 1986.

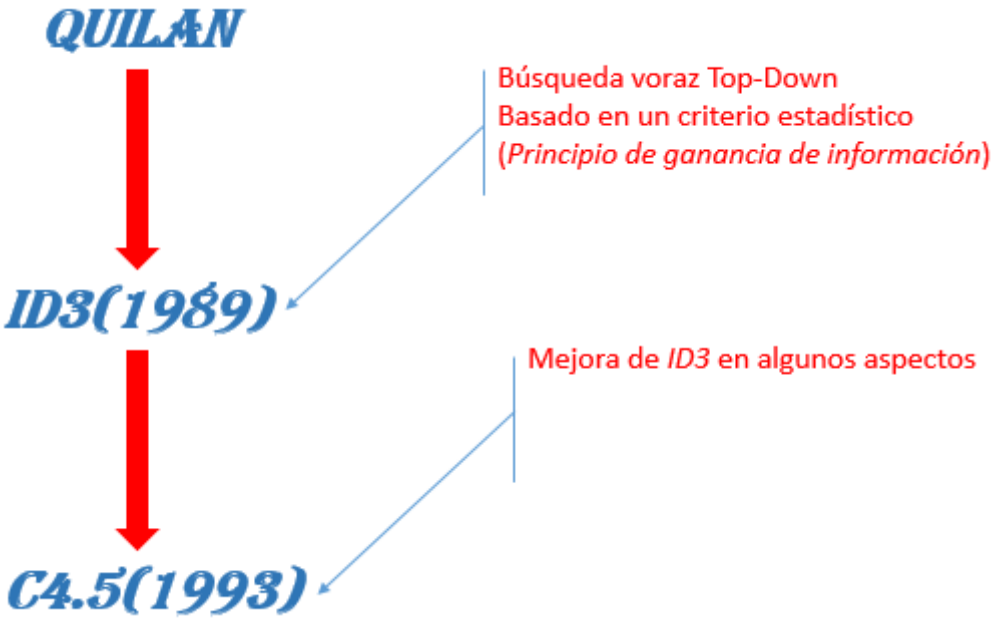


Ilustración 27. Evolución del C4.5 a partir del ID3

El algoritmo C4.5 genera un árbol de decisión a partir de los datos mediante particiones realizadas recursivamente. El árbol se construye mediante la estrategia de profundidad-primero (depth-first).

El algoritmo considera todas las pruebas posibles que pueden dividir el conjunto de datos y selecciona la prueba que resulta en la mayor ganancia de información. Para cada atributo discreto, se considera una prueba con n resultados, siendo n el número de valores posibles que puede tomar el atributo. Para cada atributo continuo, se realiza una prueba binaria sobre cada uno de los valores que toma el atributo en los datos. En cada nodo, el sistema debe decidir cuál prueba escoge para dividir los datos.

Los tres tipos de pruebas posibles propuestas por el C4.5 son

- La prueba "estándar" para las variables discretas, con un resultado y una rama para cada valor posible de la variable.

- Una prueba más compleja, basada en una variable discreta, en donde los valores posibles son asignados a un número variable de grupos con un resultado posible para cada grupo, en lugar de para cada valor.

- Si una variable A tiene valores numéricos continuos, se realiza una prueba binaria con resultados A <= Z y A > Z, para lo cual debe determinarse el valor límite Z.

Todas estas pruebas se evalúan de la misma manera, mirando el resultado de la proporción de ganancia, o alternativamente, el de la ganancia resultante de la división que producen. Ha sido útil agregar una restricción adicional: para cualquier división, al menos dos de los subconjuntos Ci deben contener un número razonable de casos. Esta restricción, que evita las subdivisiones casi triviales, es tenida en cuenta solamente cuando el conjunto C es pequeño.

3.3.1.1.3 Características del algoritmo

- Permite trabajar con valores continuos para los atributos, separando los posibles resultados en 2 ramas Ai<=N y Ai>N.

- Los árboles son menos frondosos, ya que cada hoja cubre una distribución de clases no una clase en particular.

- Utiliza el método "divide y vencerás" para generar el árbol de decisión inicial a partir de un conjunto de datos de entrenamiento.

- Se basa en la utilización del criterio de proporción de ganancia (gain ratio), definido como I(Xi,C)/H(Xi). De esta manera se consigue evitar que las variables con mayor número de posibles valores salgan beneficiadas en la selección. El cálculo de dicha proporción se procede como sigue:

Dados S (un conjunto de ejemplos de aprendizaje) y A (un atributo de los ejemplos que puede tomar c posibles valores) definimos:

$$SplitInformation(S,A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} log_2 \frac{|S_i|}{|S|}$$

SplitInformation(S, A) se denota la entropía de S con respecto a los valores de A:

$$RatioGanancia(S,A) = \frac{Ganancia(S,A)}{SplitInformation(S,A)}$$

Donde $Ganancia(S,A)$ se define como la diferencia entre la entropía de S, y la entropía de S considerando A:

$$Ganancia(S,A) = SplitInformation(S) - SplitInformation(S,A)$$

$RatioGanancia$ (S,A) favorece aquellos atributos que, en igualdad de Ganancia, separen los datos en menos clases

- Es Recursivo.

3.3.1.1.4 Heurística

Utiliza una técnica conocida como Gain Ratio (proporción de ganancia). Es una medida basada en información que considera diferentes números (y diferentes probabilidades) de los resultados de las pruebas

3.3.1.1.5 Atributos

Atributos de valores continuos: Inicialmente el algoritmo ID3 se planteó para atributos que presentaban un número discreto de valores. Podemos fácilmente incorporar atributos con valores continuos, simplemente dividiendo estos valores en intervalos discretos, de forma que el atributo tendrá siempre valores comprendidos en uno de estos intervalos.

Medidas alternativas en la selección de atributos: Al utilizar la ganancia de información estamos introduciendo involuntariamente un sesgo que favorece a los atributos con muchos valores distintos. Debido a que dividen el conjunto de ejemplos en muchos subconjuntos, la ganancia de información es forzosamente alta. Sin embargo, estos atributos no son buenos predictores de la función objetivo para nuevos ejemplos. Una medida alternativa que se ha usado con éxito es la "gain ratio".

3.3.1.1.6 Ventajas y mejoras respecto al algoritmo id3

• Evitar Sobreajuste de los datos.

• Determinar como de profundo debe crecer el árbol de decisión.

• Reducir errores en la poda.

• Condicionar la Post-Poda.

• Manejar atributos continuos.

• Escoger un rango de medida apropiado.

• Manejo de datos de entrenamiento con valores faltantes.

• Manejar atributos con diferentes valores.

• Mejorar la eficiencia computacional.

3.3.1.1.7 Sobreajuste (overfitting)

A medida que se añaden niveles AD, las hipótesis se refinan tanto que describen muy bien los ejemplos utilizados en el aprendizaje, pero el error de clasificación puede aumentar al evaluar los ejemplos. Es decir, clasifica muy bien los datos de entrenamiento pero luego no sabe generalizar al conjunto de prueba. Es debido a que aprende hasta el ruido del conjunto de entrenamiento, adaptándose a las regularidades del conjunto de entrenamiento.

Este efecto es, por supuesto, indeseado. Hay varias causas posibles para que esto ocurra. Las principales son:

• Exceso de ruido (lo que se traduce en nodos adicionales)

• Un conjunto de entrenamiento demasiado pequeño como para ser una muestra representativa de la verdadera función objetivo.

Hay varias estrategias para evitar el sobreajuste en los datos. Pueden ser agrupadas en dos clases:

• Estrategias que frenan el crecimiento del árbol antes de que llegue a clasificar perfectamente los ejemplos del conjunto de entrenamiento.

• Estrategias que permiten que el árbol crezca completamente, y después realizan una poda.

3.3.1.1.8 Post Prunning (post poda)

Es una variante de la poda y es usada por el C4.5. Consiste en una vez generado el árbol completo, plantearse qué es lo que se debe "podar" para mejorar el rendimiento y de paso obtener un árbol más corto. Pero además el C4.5 convierte el árbol a un conjunto de reglas antes de podarlo. Hay tres razones principales para hacer esto:

• Ayuda a distinguir entre los diferentes contextos en los que se usa un nodo de decisión, debido a que cada camino de la raíz a una hoja se traduce en una regla distinta.

• Deja de existir la distinción entre nodos que están cerca de la raíz y los que están lejos. Así no hay problemas para reorganizar el árbol si se poda un nodo intermedio.

• Mejora la legibilidad. Las reglas suelen ser más fáciles de entender.

3.3.1.1.9 Estructuras utilizadas en el algoritmo C4.5

El C4.5 forma parte de la familia de los TDIDT (Top Down Induction Trees), junto con antecesor el ID3.

El C4.5 se basa en el ID3, por lo tanto, la estructura principal de ambos métodos es la misma.

El C4.5 construye un árbol de decisión mediante el algoritmo "divide y vencerás" y evalúa la información en cada caso utilizando los criterios de Entropía, Ganancia o proporción de ganancia, según sea el caso.

Por otra parte, los árboles de decisión pueden entenderse como una representación de los procesos involucrados en las tareas de clasificación. Están formados por:

♦ Nodos: Nombres o identificadores de los atributos.

♦ Ramas: Posibles valores del atributo asociado al nodo.

♦ Hojas: Conjuntos ya clasificados de ejemplos y etiquetados con el nombre de una clase

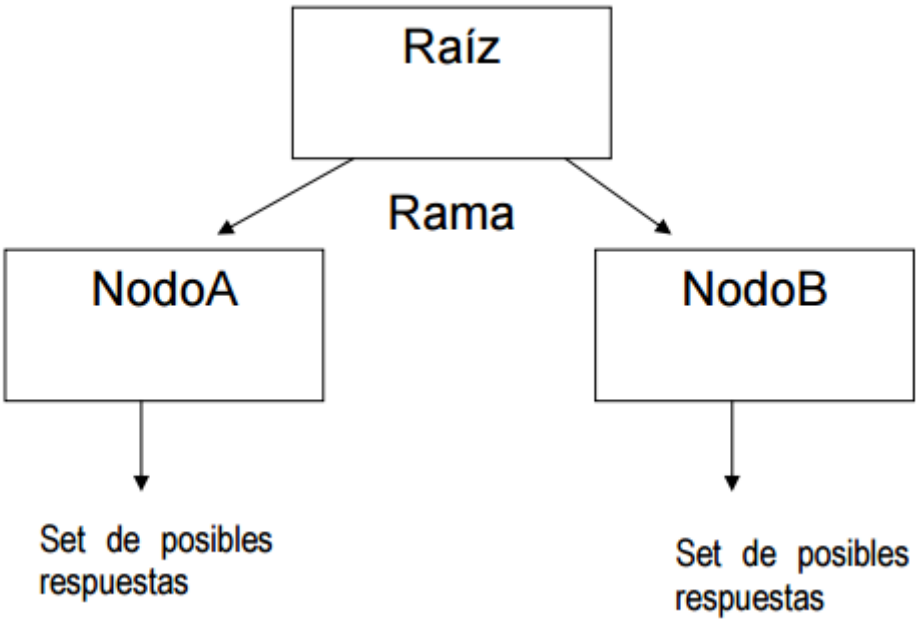


Ilustración 28. Estructura genérica de un árbol de decisión inducido por el C4.5

La ilustración 28, muestra el modelo genérico del árbol de decisión inducido por el entrenamiento del algoritmo C4.5.

3.3.1.1.10 Pseudocódigo de c4.5

Función C4.5

R: conjunto de características a entrenar (en nuestro caso las características de color F_i).

C: clase o decisión a clasificar (en nuestro caso será velo o no velo),

S: conjunto de entrenamiento, o las muestras de entrenamiento (en nuestro caso son 150 muestras de píxeles cada una tiene sus 15 características)

Comienzo

Si S está vacío,

Devolver un único nodo con Valor Falla; ‘para formar el nodo raíz

Si todos los registros de S tienen el mismo valor para el atributo clasificador,

Devolver un único nodo con dicho valor; ‘un único nodo para todos

Si R está vacío,

Devolver un único nodo con el valor más frecuente del atributo

Clasificador en los registros de S [Nota: habrá errores, es decir,

Registros que no estarán bien clasificados en este caso];

Si R no está vacío,

D ←Atributo con mayor Proporción de Ganancia (S,D) entre los atributos de R;

Sean {dj | j=1,2,...., m} los valores del atributo D;

Sean {Sj | j=1,2,...., m} los subconjuntos de S correspondientes a los valores de dj respectivamente del atributo D;

Devolver un árbol con la raíz nombrada como D y con los arcos nombrados S1, S2,...., Sm, que van respectivamente a los árboles, con las características R computada en computadas en los nodos, y donde C es la clase:

C4.5(R-{D}, C, Sl), C4.5(R-{D}, C, S2),…,C4.5(R-{D}, C, Sm);

Fin

3.3.1.1.11 Diagrama genérico del algoritmo c4.5

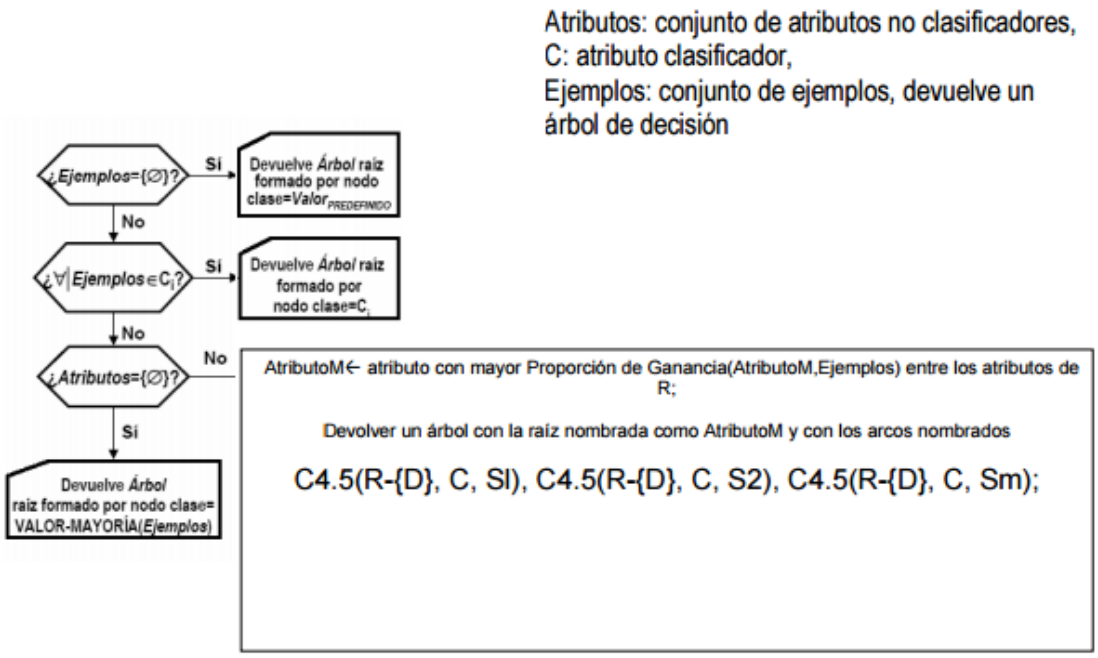


Ilustración 29. Diagrama genérico del algoritmo C4.5

3.3.1.1.12 Estimación de la Proporción de Errores

Una vez podados, las hojas de los árboles de decisión generados por el C4.5 tendrán dos números asociados: N y E. N es la cantidad de casos de entrenamiento cubiertos por la hoja, y E es la cantidad de errores predichos si un conjunto de N nuevos casos fuera clasificados por el árbol. La suma de los errores predichos en las hojas, dividido el número de casos de entrenamiento, es un estimador inmediato del error de un árbol podado sobre nuevos casos.

El C4.5 es una extensión del ID3 que acaba con muchas de sus limitaciones. Por ejemplo, permite trabajar con valores continuos para los atributos, separando los posibles resultados en dos ramas: una para aquellos Ai<=N y otra para Ai>N. Además, los árboles son menos frondosos porque cada hoja no cubre una clase en particular sino una distribución de clases, lo cual los hace menos profundos y menos frondosos. Este algoritmo fue propuesto por Quinlan en 1993. El C4.5 genera un árbol de decisión a partir de los datos mediante particiones realizadas recursivamente, según la estrategia de profundidad-primero (depth-first). Antes de cada partición de datos, el algoritmo considera todas las pruebas posibles que pueden dividir el conjunto de datos y selecciona la prueba que resulta en la mayor ganancia de información o en la mayor proporción de ganancia de información. Para cada atributo discreto, se considera una prueba con n resultados, siendo n el número de valores posibles que puede tomar el atributo. Para cada atributo continuo, se realiza una prueba binaria sobre cada uno de los valores que toma el atributo en los datos.

Ahora que hemos definido teóricamente el algoritmo C4.5, vamos a aplicarlo sobre nuestro proyecto con el fin de entrenar las características de color de los píxeles de lesiones, y obtener un árbol de decisión clasificador. Para ello, vemos las funciones implementadas en Matlab en el siguiente punto.

3.3.1.2 Implementación Matlab

A esta altura del desarrollo del proyecto, lo que nos queda por hacer es entrenar nuestro conjunto de características que hemos obtenido en la sección 3.2.3, para obtener nuestro árbol de decisión. Pero antes vamos nuestra función de Matlab que ejecuta el algoritmo C4.5.

En efecto disponemos de dos funciones:

La primera se llama **C4_5.m**, que a su vez llama a otra función llamada **make_tree.m**.

C4_5.m va a ser como la función principal, pero en realidad la función que construye el árbol de decisión, es decir la que aplica el algoritmo C4.5 va ser **make_tree.m.**

El diagrama de flujos de la función **C4_5.m** se ilustra en la siguiente ilustración 30.

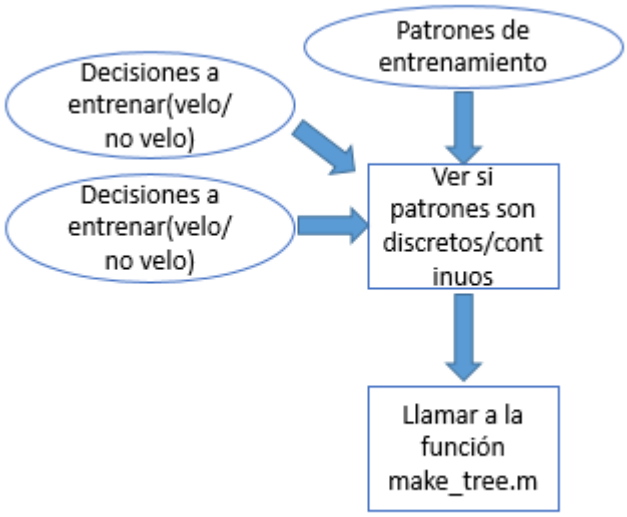


Ilustración 30. Diagrama de flujos de la función **C4_5.m**

El código Matlab de la función **C4_5.m** es:

```
function  [tree]= C4_5(train_patterns, train_targets,inc_node)
% Classicación usando el algoritmo C4.5 de Quinlan
% Entradas:
%   training_patterns    - Los atributos a entrenar
%   training_targets     - Las decisiomes a entrenar
%   inc_node             - Porcentaje de los ejemplos incorrectos asignados a
un nodo
% Salida:
%    Tree - El árbol
%NOTE: En esta implementación, se supone que un vector patrón con menos de 10
valores únicos (el parámetro Nu)
% Es discreto, y será tratada como tal. En otro caso será tratado como
continuo
[Ni, M]      = size(train_patterns);
inc_node     = inc_node*M/100;
Nu           = 10;
%Encontrar cuál de los patrones de entrada son discretos

discrete_dim = zeros(1,Ni);
for i = 1:Ni,
 Ub = unique(train_patterns(i,:));
 Nb = length(Ub);
  if (Nb <= Nu),
   %  Es un patrón discreto
      discrete_dim(i)    = Nb;

 end
end
```

```
%Contruir recursivamente el árbol
disp('Building tree')
tree            = make_tree(train_patterns, train_targets, inc_node,
discrete_dim, max(discrete_dim), 0);
end
```

Tabla 7. Código Matlab de la función **C4.5.m**

A continuación describimos la función **C4_5.m**:

Esta función digamos que es como la función "main" ya que desde ella se hace la llamada a la función que realmente nos construye el árbol que es **make_tree.m** y que vamos a ver después. Pero antes de hacer la llamada a **make_tree.m**, realiza el paso de verificar si nuestra matriz de patrones contiene valores discretos o continuos de las características a entrenar.

Las entradas de esta función son la variable "train_patterns" que es donde hemos guardado las características de color de las 150 muestras (recortes) de entrenamiento. "train_targets" es entonces una matriz 15x150 es decir de 15 filas y 150 columnas, cada fila representa una característica y contiene los valores promedios de esa característica para todos los recortes 150 en total, y cada columna representa una muestra (recorte) con los valores promedios de sus 15 características. La ilustración 31 nos muestra la estructura de esta matriz donde se pueden ver las 15 filas pero solo hasta la columna muestra 17 por el número alto de muestras (150) que no caben en un pantallazo.

PAT <15x150 double>

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0.2727	0.3700	1	0	0.1100	0.2619	0	0	0.3400	0.7460	0	0.3281	0	0.0251	0.2194	0	0.0386	
2	0.0818	0.0300	0.9876	0	0.0100	0	0	0.0064	0.3600	0.8289	0	0.2813	0	0.0082	0.2271	0.0125	0.0847	
3	0.5545	0.8300	1	0.5247	0.7700	0.4286	0	0.0321	0.3000	0.9365	0	0.3625	0	0.0117	0.2852	0.0821	0.1042	
4	1	1	1	0	0.4000	0.6429	0	0	0.4000	0.9859	0	0.4781	0	0.1340	0.2335	7.3260e-04	0.0577	
5	1	1	1	0.6852	0.8400	0.7143	0	0.5705	0.5100	1	0	0.4938	0.0026	0.0723	0.3032	0.0352	0.2556	
6	1	1	1	1	1	0.9286	0	0.8590	0.6700	1	0	0.6719	0	0.4866	0.5381	0.2989	0.7714	
7	0.3333	0.3333	0.3333	0	0.1786	0.2813	NaN	0	0.2532	0.3302	NaN	0.2909	0	0.1934	0.2173	0.0022	0.0532	
8	0.3333	0.3333	0.3333	0.4066	0.3750	0.3125	NaN	0.3991	0.3228	0.3349	NaN	0.3004	1	0.1043	0.2821	0.1050	0.2356	
9	0.3333	0.3333	0.3333	0.5934	0.4464	0.4063	NaN	0.6009	0.4241	0.3349	NaN	0.4087	0	0.7023	0.5006	0.8928	0.7112	
10	-70.6934	-67.6289	-93.9930	-147.3426	-119.2213	-112.6042	-175.7342	-136.5508	-112.7403	-81.7679	-151.2517	-114.8037	-159.2663	-129.5589	-128.2843	-167.9427	-146.6581	-9
11	-41.2575	-41.2547	-69.1743	-96.0210	-90.9632	-93.4960	-135.2586	-96.9905	-86.5518	-58.5858	-138.7977	-99.5901	-142.8674	-122.4054	-98.5814	-146.1479	-114.1404	-7
12	4.8660	9.9469	-12.6063	-52.6643	-53.6277	-69.7815	-119.7657	-61.5368	-68.4459	-29.7216	-138.5609	-63.5126	-132.0051	-93.9360	-67.6421	-109.1732	-93.3853	-5
13	0.6602	0.6836	0.5347	0.4977	0.4519	0.4082	0.4080	0.4628	0.4211	0.4808	0.3529	0.4131	0.3669	0.3746	0.4356	0.3968	0.4141	
14	0.3853	0.4170	0.3935	0.3244	0.3448	0.3389	0.3140	0.3287	0.3233	0.3445	0.3238	0.3584	0.3291	0.3539	0.3347	0.3453	0.3223	
15	-0.0454	-0.1005	0.0717	0.1779	0.2033	0.2529	0.2780	0.2085	0.2556	0.1748	0.3233	0.2285	0.3041	0.2716	0.2297	0.2579	0.2637	

Ilustración 31. Trozo de la matriz PAT que contiene los atributos de entrenamiento

La segunda entrada de la función **C4_5.m** es la variable "train_targets", que es un vector fila conteniendo las dos posibles decisiones o clases, que en nuestro caso hay dos: velo o no velo, hemos dado el valor lógico 1 (verdadero para velo) y el valor cero (falso) para el caso de no velo. Como tenemos en total 150 muestra de entrenamiento 60 velo y 90 no velo, "train_targets" será un vector fila de 150 posiciones, donde las 60 primeras posición son de valor 1 (velo) y las siguientes 90 posiciones son de valor 0 (no velo).

Notar que tanto la función **C4_5.m** como **make_tree.m** se han realizado de tal forma que respetan la estructura descrita de "train_patterns" y "train_targets" a la hora de recorrerlos.

La tercera entrada de **C4_5.m:** "inc_node", es simplemente el porcentaje de error permitido de asignación a un nodo, que después en la línea 16 se va a multiplicar por el número de muestras partido por 100 (M/100). Cuanto más grande es este valor más rango de errores le dejamos a nuestro algoritmo, para nuestro árbol le hemos dado el valor 0.35, que es un valor que nos deja un rango de errores más o menos pequeño, o sea más preciso sale el árbol. Por defecto en Matlab está limitado el número de iteraciones en una función recursiva, pues si le pasamos un valor de inc_node menor que 0.30 nos sale un error de límite alcanzado a la hora el ejecutar las funciones.

Como salida nos devuelve la variable "tree", que contendrá la información del árbol

Como ya se ha mencionado, lo que hace la función es básicamente diferenciar los patrones discretos de los continuos, mediante la variable discrete_dim que será un vector de ceros si el patrón es discreto y distinto de cero si es continuo. Después se hace la llamada a **make_tree.m** dándole como parámetros "train_patterns", "train_targets", "inc_node", y "discrete_dim", así como su valor máximo "max (discrete_dim)", función que vamos a ver a continuación:

El diagrama de flujos de la función **make_tree.m** es en efecto el del algoritmo C4.5 genérico que se ilustra en la ilustración 31, salvo que nuestra función hace los cálculos por separado para los patrones discretos y para los continuos, siguiendo la misma lógica general del algoritmo C4.5.

El código Matlab de la función **make_tree.m** es:

```
function tree = make_tree(patterns, targets, inc_node, discrete_dim, maxNbin,
base)


[Ni, L]                     = size(patterns);
Uc                          = unique(targets);
tree.dim                    = 0;
tree.split_loc              = inf;
if isempty(patterns),
    return
end
%Cuando parar: Si la dimensión es uno o el número de muestras es pequeño
if ((inc_node > L) | (L == 1) | (length(Uc) == 1)),
    H                   = hist(targets, length(Uc));
    [m, largest]    = max(H);
    tree.Nf         = [];
    tree.split_loc  = [];
    tree.child      = Uc(largest);
    return
end
%Computar el nodo I
for i = 1:length(Uc),
    Pnode(i) = length(find(targets == Uc(i))) / L;
end
Inode = -sum(Pnode.*log(Pnode)/log(2));
%Para cada dimensión, calcular la impureza relación de ganancia
% Esto se hace por separado para los patrones discretos y continuos
delta_Ib    = zeros(1, Ni);
split_loc   = ones(1, Ni)*inf;
for i = 1:Ni,
    data    = patterns(i,:);
    Ud      = unique(data);
    Nbins   = length(Ud);
    if (discrete_dim(i)),
        %Es un patrón discreto
        P   = zeros(length(Uc), Nbins);
        for j = 1:length(Uc),
            for k = 1:Nbins,
                indices     = find((targets == Uc(j)) & (patterns(i,:) ==
Ud(k)));
                P(j,k)  = length(indices);
            end
        end
        Pk          = sum(P);
        P           = P/L;
        Pk          = Pk/sum(Pk);
        info        = sum(-P.*log(eps+P)/log(2));
        delta_Ib(i) = (Inode-sum(Pk.*info))/-sum(Pk.*log(eps+Pk)/log(2));
    else
        %Es un patrón continuo
        P   = zeros(length(Uc), 2);
          %Sortear los patrones
          [sorted_data, indices] = sort(data);
          sorted_targets = targets(indices);
          %Calcular la información para cada posible ramificación
          I = zeros(1, L-1);
          for j = 1:L-1,
              P(:, 1) = hist(sorted_targets(1:j) , Uc);
              P(:, 2) = hist(sorted_targets(j+1:end) , Uc);
              Ps        = sum(P)/L;
```

```
                P      = P/L;
                Pk       = sum(P);
                P1       = repmat(Pk, length(Uc), 1);
                P1       = P1 + eps*(P1==0);
                info   = sum(-P.*log(eps+P./P1)/log(2));
                I(j)   = Inode - sum(info.*Ps);
            end
            [delta_Ib(i), s] = max(I);
            split_loc(i) = sorted_data(s);
        end
    end
%Hallar la dimensión minimizando delta_Ib
[m, dim]     = max(delta_Ib);
dims         = 1:Ni;
tree.dim     = dim;
%Ramificar a lo largo de 'dim': la dimensión
Nf       = unique(patterns(dim,:));
Nbins    = length(Nf);
tree.Nf = Nf;
tree.split_loc       = split_loc(dim);
%Si sólo un valor pertenece a este patrón, ya no hay más ramas.
if (Nbins == 1)
    H                = hist(targets, length(Uc));
    [m, largest]     = max(H);
    tree.Nf          = [];
    tree.split_loc   = [];
    tree.child       = Uc(largest);
    return
end
if (discrete_dim(dim)),
    %Patrón discreto
    for i = 1:Nbins,
        indices          = find(patterns(dim, :) == Nf(i));
        tree.child(i)    = make_tree(patterns(dims, indices),
targets(indices), inc_node, discrete_dim(dims), maxNbin, base);
    end
else
    %Patrón continuo
    indices1             = find(patterns(dim,:) <= split_loc(dim));
    indices2             = find(patterns(dim,:) > split_loc(dim));
    if ~(isempty(indices1) | isempty(indices2))
        tree.child(1)    = make_tree(patterns(dims, indices1),
targets(indices1), inc_node, discrete_dim(dims), maxNbin, base+1);
        tree.child(2)    = make_tree(patterns(dims, indices2),
targets(indices2), inc_node, discrete_dim(dims), maxNbin, base+1);
    else
        H                = hist(targets, length(Uc));
        [m, largest]     = max(H);
        tree.child       = Uc(largest);

        tree.dim                 = 0;
    end
end
end
```

Tabla 8. Código Matlab de la función **make_tree.m**

A continuación describimos la función **make_tree.m**:

La función **make_tree.m** es una función recursiva es decir que vuelve a llamar a sí misma, ya que puede haber muchos nodos y ramas en el árbol resultante. Los parámetros de entrada de la función ya los hemos descrito antes, también es el caso de la salida que es la misma variable "tree". Esta variable que es el árbol inducido, va a ser una estructura cuyos campos nos darán la información del árbol de decisión.

Primero se guardan las filas (características de color), y las columnas (muestras o recortes) en las variables auxiliares "Ni" y "M", para su recorrido posterior. Se declaran también los campos de la estructura "tree". El primer campo es "dim" ("tree.dim"), donde vamos a tener la característica de color del nodo raíz y los otros nodos del árbol de las posibles ramas, en nuestro caso va a ser un valor entre 1 y 15 ya que es el número de características a clasificar. El otro campo de la estructura la variable "split_loc" ("tree.split_loc") que será el valor de la característica de color a comparar en los nodos. El tercer campo de la estructura, no tiene importancia para construir el árbol es "Nf" ("tree.Nf"), nos dará los valores mínimos y máximos del vector línea de la característica del campo dim, que se ha computado. Y finalmente el último campo de la estructura es "child" ("tree.child"), que será el hijo, de un nodo, es decir su ramificación, que realmente es una sub estructura con los mismos campos descritos antes (En la sección 3.3.2 se va a explicar mejor la estructura tree obtenida).

La función comprueba primero si la matriz de entrenamiento está vacía, si no lo continua. Después se hace una comprobación de que si hay un mínimo número de información a entrenar o si el rango de errores (variable "inc_node") es mayor que el número de muestras, en caso afirmativo se para la función dando valores vacíos a los campos de "tree", si no se continúa con el algoritmo.

A continuación se empieza a computar los nodos a crear comenzando naturalmente por el nodo raíz y así sucesivamente. Recorriendo las características de color, se calcula la relación de ganancia después de sortear los patrones. Esto se hace por separado para los patrones discretos y continuos, y se hace el cálculo por cada posible ramificación. Después se identifica la característica del nodo computado, es decir el campo "dim" de la estructura del árbol, maximizando la ganancia. También se halla el valor límite de la característica del nodo en cuestión.

No hay más ramas cuando hay solo un valor perteneciente a la característica computada a comparar, eso es que la variable "Nbins" es igual a uno. Entonces se para el árbol, dando valores vacíos/nulos a los campos de la estructura. En caso contrario si "Nbins" es mayor que 1, se vuelve a llamar a la función **make_tree.m** recursivamente, creando por lo tanto un nuevo campo hijo "child" de la estructura "tree", que será una nueva rama del árbol. Otra vez esto último se hace tanto para los patrones discretos como los continuos de forma separada.

Entonces desde la ventana de comandos de Matlab hacemos la llamada a **C4_5.m** para crear nuestro árbol de decisión, pasándole como parámetros la matriz de las características extraídas de los píxeles de entrenamiento que hemos guardado en la variable "PAT", y el vector línea de decisiones velo/no velo, que hemos guardado en la variable "TARG", y el valor 0.35 de inc_node, tal como sigue:

```
>> tree=C4_5(PAT,TARG,0.35);

Building tree

>>
```

Después del display "Building tree" que tarda unos segundos, se guarda la estructura del árbol en la variable "tree". En el siguiente punto vamos a describir esta estructura y deducir el árbol de decisión a partir de ella.

3.3.2 Árbol de decisión inducido

En este apartado vamos a obtener el árbol de decisión inducido por al algoritmo C4.5, después de entrenarlo en el apartado anterior con la matriz de atributos de entrenamiento PAT más el vector de decisiones TARG.

Obtenemos por lo tanto la variable "tree" que guarda la estructura del árbol. Tiene este aspecto cuando lo abrimos en el "workspace" de Matlab:

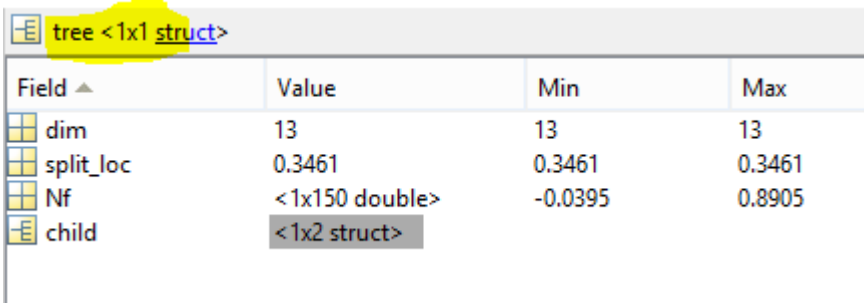


Ilustración 32. Aspecto de la estructura tree nodo raíz

La ilustración 32 muestra los diferentes campos de la estructura tree. Vemos que en el nodo raíz se ha computado la característica F13 ya que el campo "dim" tiene valor 13, con el valor límite 0.3461, en el campo "split_loc". También podemos ver que tiene un hijo guardado en el campo "child", que como se puede ver contiene dos subestructuras o ramas. Al pulsar sobre el campo "child", se abre la siguiente pestaña (ilustración 33).

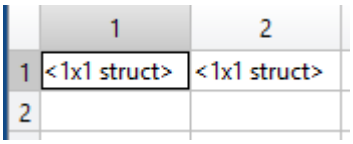


Ilustración 33. Pestaña del campo “child” de la estructura tree

Podemos ver las dos ramificaciones o sub estructuras. Cuando abrimos la primera sale:

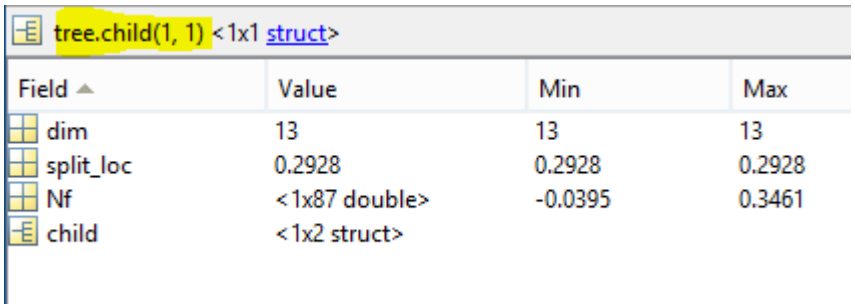
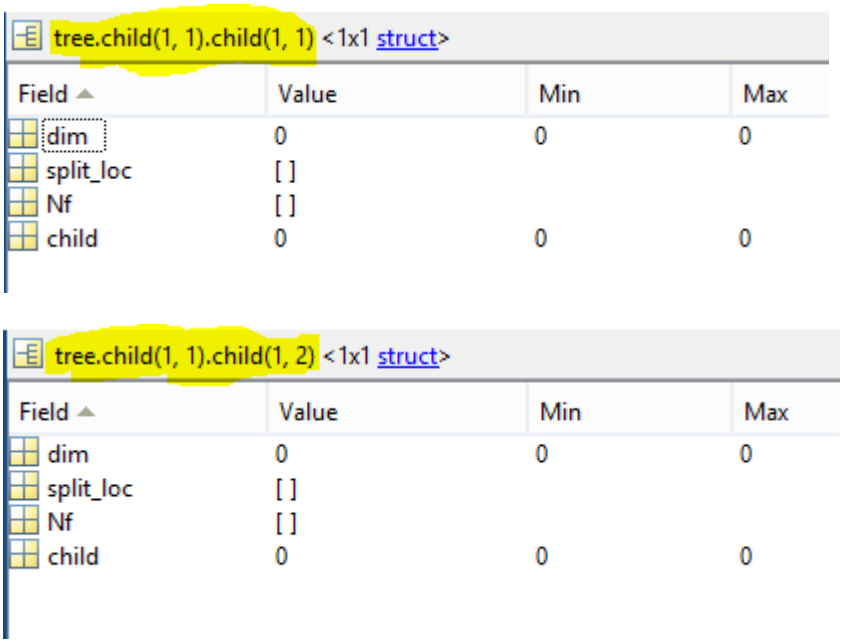


Ilustración 34. Aspecto de la sub estructura tree.child (1,1)

Podemos observar en la ilustración 34 que se computa la misma característica pero con un valor menor que le anterior 0.2928. Puesto que al verificar nuestra matriz de características “PAT” de las 150 muestras entrenadas, no había ningún valor de la característica F13 entre los valores 0.2928 y 0.3461, eso es que nuestro algoritmo ha hecho una mejora de del nodo raíz, por lo que vamos a sustituir el nodo anterior por este último.

Si abrimos el campo child de esta subestructura encontramos valores nulos:



tree.child(1, 1).child(1, 1) <1x1 struct>

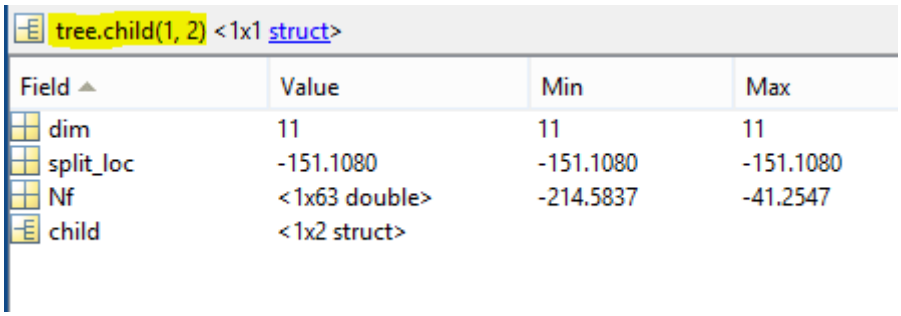
Field	Value	Min	Max
dim	0	0	0
split_loc	[ ]		
Nf	[ ]		
child	0	0	0

tree.child(1, 1).child(1, 2) <1x1 struct>

Field	Value	Min	Max
dim	0	0	0
split_loc	[ ]		
Nf	[ ]		
child	0	0	0

Ilustración 35. Aspecto de las dos ramas de la sub estructura tree.child (1,1)

Por lo tanto se para el árbol aquí no hay más ramas. La segunda subestructura de "tree" tree.child (1,2) tiene este aspecto:

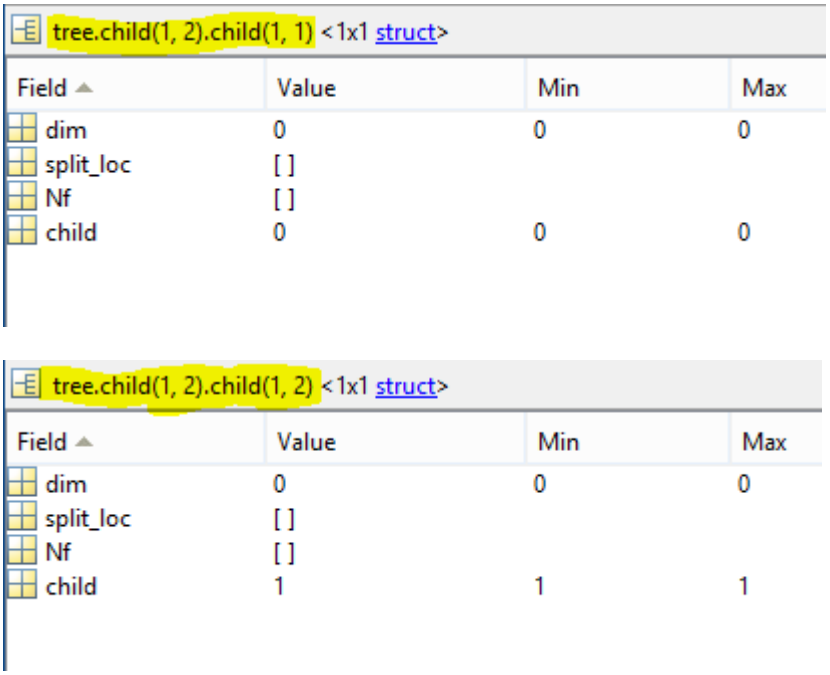


tree.child(1, 2) <1x1 struct>

Field	Value	Min	Max
dim	11	11	11
split_loc	-151.1080	-151.1080	-151.1080
Nf	<1x63 double>	-214.5837	-41.2547
child	<1x2 struct>		

Ilustración 36. Aspecto de la sub estructura tree.child (1,2)

De la ilustración 36 deducimos que va a haber una rama colgando de tree.child (1,1), que computa la característica F11 con el valor límite -151.1080. Vamos a ver que las dos subestructuras que cuelgan de esta rama son nulas al igual que en el caso anterior:



tree.child(1, 2).child(1, 1) <1x1 struct>

Field	Value	Min	Max
dim	0	0	0
split_loc	[ ]		
Nf	[ ]		
child	0	0	0

tree.child(1, 2).child(1, 2) <1x1 struct>

Field	Value	Min	Max
dim	0	0	0
split_loc	[ ]		
Nf	[ ]		
child	1	1	1

Ilustración 37. Aspecto de las dos ramas de la sub estructura tree.child (1,2)

Por consiguiente y de igual modo que en el caso anterior, se para el árbol en este punto.

Ahora que tenemos claro el tamaño de nuestro árbol, nos falta solamente un último paso, que es ver si los valores de los campos "split_loc" 0.2928 y -151.1080 son máximo o mínimos con comparación a las características F13 y F11 respectivamente, y por último ver las decisiones que resultan de estos nodos si son velo o no velo. Para ello hay que recurrir a nuestra matriz PAT que contiene los atributos de entrenamiento e ir comparando muestra a muestra fijándonos en las características F13 y F11. Las decisiones son obvias ya que se sabe de antemano que las 60 primeras son muestras de píxeles de velo, y las siguientes 90 columnas son muestras de píxeles de no velo.

Después de realizar esta tarea, se podía llegar de manera obvia a la siguiente estructura del árbol de decisión, y que va a ser nuestro árbol de decisión definitivo para la clasificación de píxeles de nuestro proyecto:

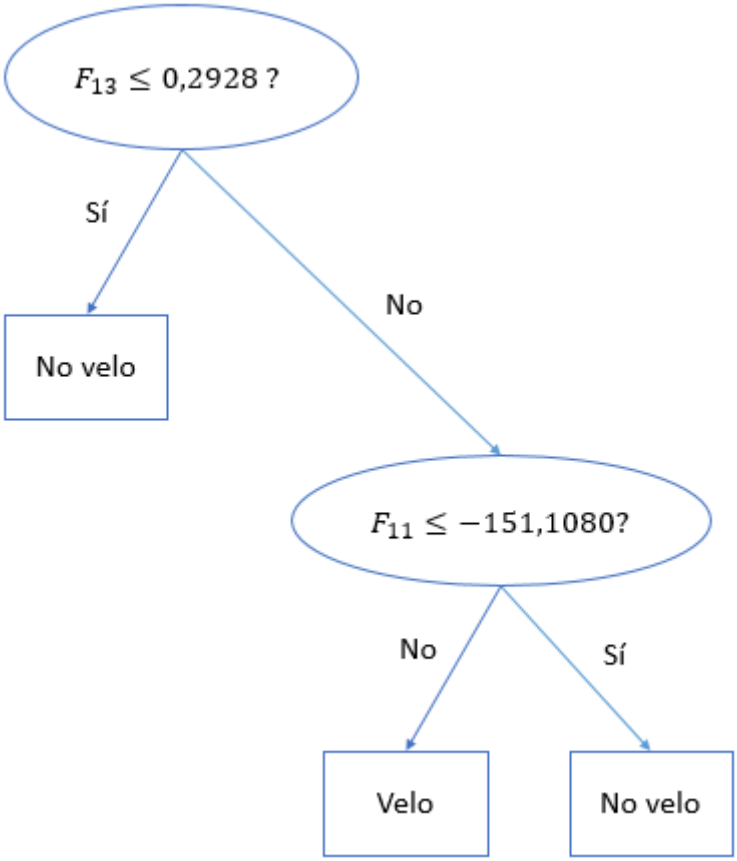


Ilustración 38. Árbol de decisión clasificador de píxeles

La ilustración 38 nos muestra el árbol de decisión inducido. Se puede observar que tan sólo 2 de las 15 características se incluyeron en el modelo de clasificación creado por el algoritmo C4.5. Las dos son características de color relativo F11 y F13.

En el siguiente apartado vamos a aplicar el clasificador de píxeles obtenido, sobre nuestro conjunto de imágenes con el fin de probarlo y así detectar las regiones de velo de manera automática.

3.3.3 Detección de regiones de velo

Este apartado tiene especial importancia pues es la aplicación de todo lo que se ha realizado anteriormente. Como ya sabemos, en la fase de entrenamiento clasificador, 15 características fueron extraídas de los píxeles de formación. Por el contrario, en la fase de aplicación de la regla, sólo las dos características que aparecen en el árbol de decisión, a saber, F13 y F11, necesitan ser extraído de los píxeles. Por lo que vamos a implementar en Matlab la última función de este proyecto que va a ser la que detecta las

regiones de velo de las imágenes dermatoscópicas que disponemos. Esta función la llamamos **detec_reg_velo.m,** su diagrama de flujos se muestra en la ilustración 39.

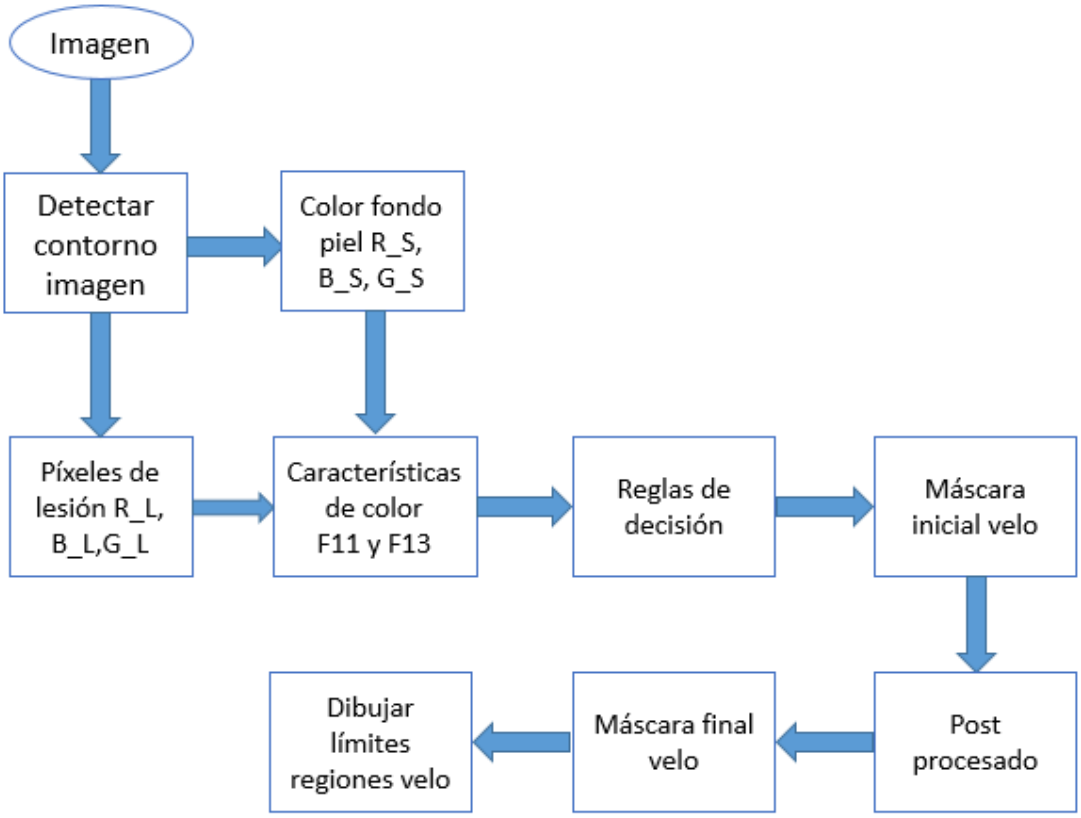


Ilustración 39. Diagrama de flujos de la función **detec_reg_velo.m**

El código Matlab de la función anterior es:

```
function []= detec_reg_velo(Imagen)

%1-->Calculo de valores RGB lesión y RGB color fondo piel
%Recorte de las imágenes para eliminar el viñeta negra que viene en el
atlas
Im_rec= imcrop(Imagen,[34 25 718 466]);
%Obtenemos la máscara del contorno de la lesión
g=detec_cont(Imagen);
%calculamos las matrices R,G,B de la imagen
Im_R=Im_rec(:,:,1);
Im_G=Im_rec(:,:,2);
Im_B=Im_rec(:,:,3);
%obtenemos la imagen en RGB, con la máscara de la lesión
R_L=uint8(g).*Im_R;
G_L=uint8(g).*Im_G;
B_L=uint8(g).*Im_B;
%obtenemos los valores de color de fondo de la piel
```

```
[R_S,B_S,G_S]= detec_color_BG(g,Imagen);


%2-->Obtención de la máscara inicial de regiones de Velo
%diferencia R,G,B relativa F11
F11=double(G_L)-G_S;
%diferencia R,G,B relativa normalizada F13
F10=double(R_L)-R_S;
F12=double(B_L)-B_S;
F13=F10./(F10+F11+F12);
%aplicación de las reglas de decisión
R1=(F13>0.2928);
R2=(F11>-151.1080);
I_veil=R1.*R2;
imshow(I_veil);


%3-->Post procesado, usando Filtro de mayoría 5x5
[X Y]=size(I_veil);
for i=3:(X-2) %Bucle para recorrer las filas de la imagen
  for j=3:(Y-2) %Bucle para recorrer las columnas de la imagen
     R=I_veil(i-2:i+2,j-2:j+2)==1;
      [r c]=find(R);
      if length(r)>=13
          I_veil(i,j)=1;
          else
          I_veil(i,j)=0;
      end
  end
end
F_veil=I_veil;
figure,imshow(F_veil);



%4-->Dibujar las los límites de las regiones de velo sobre las imágenes
dermatoscópicas
p=bwperim(F_veil);
[row,col]=find(p);
l=length(row);
for k=1:l
    Im_rec(row(k),col(k),1)=255;
    Im_rec(row(k),col(k),2)=255;
    Im_rec(row(k),col(k),3)=255;
end
figure,imshow(Im_rec);

end
```

Tabla 9. Código Matlab de la función **detec_reg_velo.m**

A continuación describimos la función:

La función **detec_reg_velo.m** es una función muy importante en este proyecto, porque es la nos detecta las regiones de velo azul-blanco, que es el objetivo final de este PFC.

Tiene un solo parámetro de entrada, que es la imagen dermatoscópica. No tiene parámetros de salida pues las regiones de velo detectadas nos las dibuja dentro de la función.

La función tiene 4 partes principales. En la primera parte se obtienen las matrices R_L, B_L y G_L, que son los valores del espacio RGB de los píxeles de la lesión, y que vamos a necesitar para calcular las matrices de características de color F11 y F13. Para obtener R_L, B_L y G_L, multiplicamos los valores RGB de la imagen entera por la máscara binaria de la lesión, así obtenemos solos los píxeles dentro de la lesión pigmentada. En esta parte también se hace una llamada a la función **detec_color_BG.m** para obtener el promedio color de la piel del fondo (R_S, B_S, G_S), necesario también para el cálculo de características de color F11, y F13.

La segunda parte de la función obtiene la máscara inicial de las regiones de velo, aplicando el la regla de decisión del árbol clasificador que hemos obtenido. Para ello se deben calcular las características de color relativas F11 y F13, necesarias en el árbol. Una vez calculadas nos fijamos en el árbol y seguimos la ruta que nos llegue a la decisión velo, como se puede ver en la ilustración 40.

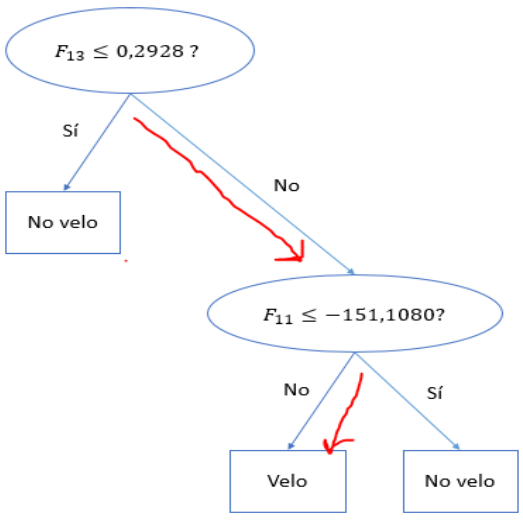


Ilustración 40. Ruta de la decisión Velo

Siguiendo la ruta anterior, que nos llega a la decisión "velo", vemos que hay dos decisiones "No", para responder a las decisiones de los dos nodos, por lo que en nuestro código de Matlab creamos un matriz lógica en la variable "I_veil", que es la multiplicación de dos matrices lógicas "R1" y "R2", que son la negación de los nodos del árbol o sea R1= (F13>0.2928) y R2= (F13>-150.1080). La matriz resultante

"I_veil", va a ser por lo tanto la máscara binaria inicial de las regiones de velo. La ilustración 41 muestra la máscara de velo inicial para la misma imagen ejemplo de la ilustración 13.

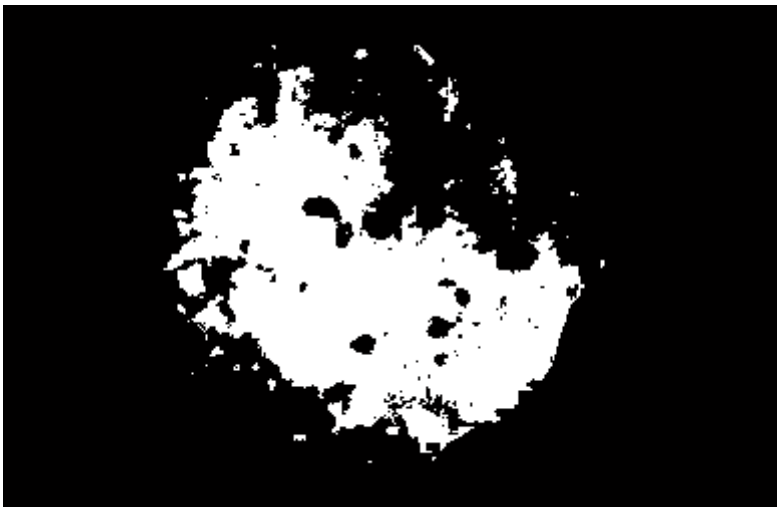


Ilustración 41. Máscara binaria inicial de regiones de velo

La tercera parte del código de la función es el post procesado. Pues como sabemos en la parte 2 de la función, para cada imagen una máscara de velo binario inicial fue generada como resultado de la aplicación de la regla. Para suavizar los bordes, un filtro mayoría 5 × 5 [23] se aplicó a las máscaras iniciales. Este filtro reemplaza cada valor de píxel con la etiqueta de la clase mayoritaria en su vecindario 5 × 5. Para ello se recorre la imagen binaria de la máscara inicial empezando por el tercer píxel tanto para recorrer filas como para recorrer columnas, para evitar un posible desbordamiento. Después para cada píxel que vale uno se verifica que sus vecinos 5x5 tengan valores unos también, son mayoría, entonces se les da valor uno, si no se les da valor nulo. La ilustración 42 muestra la máscara de velo final "F_veil", después de aplicar el filtro de mayoría 5x5. Se puede observar la diferencia entre la ilustración 41 y 42, con un claro suavizado de los bordes.

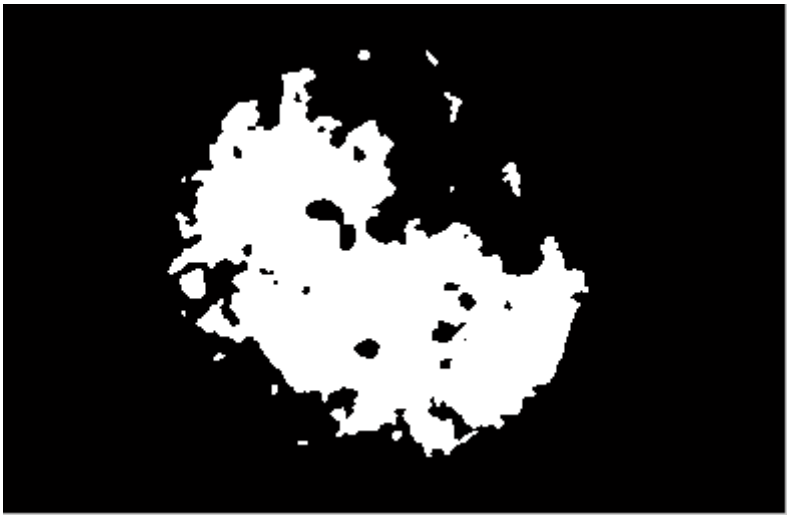


Ilustración 42. Máscara binaria final de regiones de velo

Finalmente la cuarta parte de la función **detec_reg_velo.m** es dibujar sobre las imágenes dermatoscópicas los bordes de las regiones de velo azul blanco detectados de la máscara final. Para ello convertimos esta mascara binaria final en un perímetro mediante orden *bwperim,* y lo guardamo sen la variable "p". Después mediante la orden *find*, se saben las posiciones de estos bordes, que los dibujamos en color blanco (255) sobre las imágenes originales recorriendolos. La ilustración 43 nos muestra como quedan detectadas las regiones de velo azul-blanco de la misma imagen de ejemplo.

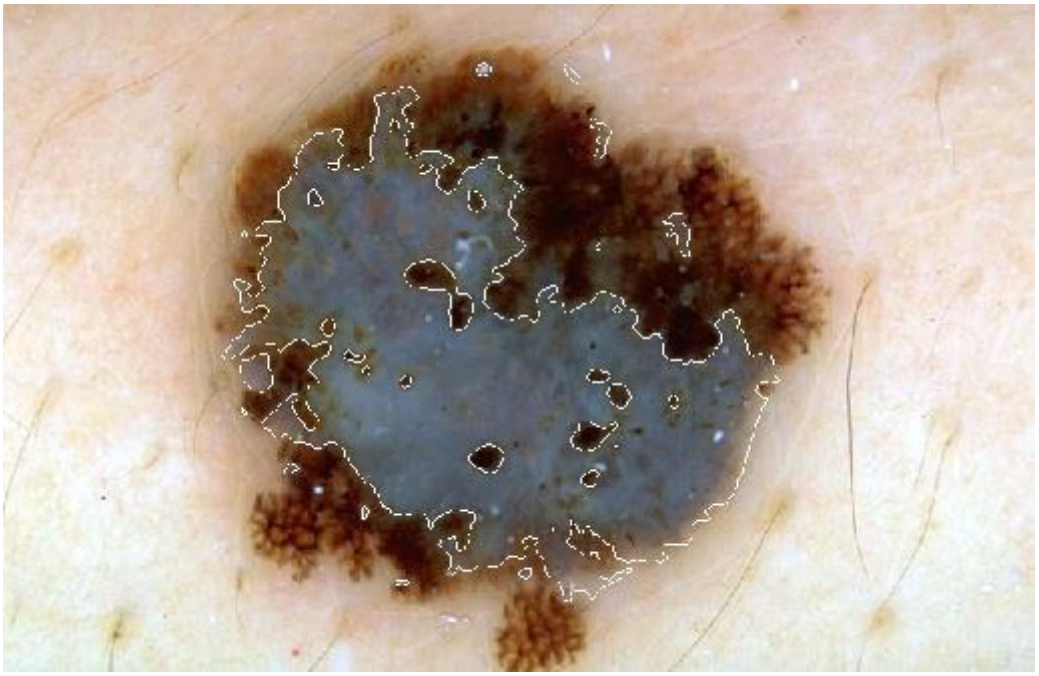


Ilustración 44. Resultado de detección de velo azul-blanco

Con esto damos por terminado el desarrollo de nuestro algoritmo de detección de velo Azul-blanco en lesiones pigmentadas de la piel. Con el fin de evaluar la eficacia del modelo de clasificación, las reglas de decisión inducidos se aplicaron a todo el conjunto de imágenes. En el siguiente capítulo vamos a resumir los resultados obtenidos en cuanto a sensibilidad y especificidad como resultado de probar las muestras de píxeles de prueba con el árbol y compararlos con los resultados obtenidos en el artículo base de este proyecto. También presentaremos la detección de velo de varias imágenes de nuestro conjunto.

CAPÍTULO 4: Resultados

En esta parte del documento, se muestran los resultados obtenidos en nuestro estudio. Para ello se utilizó la base de datos de 90 imágenes dermatoscópicas que disponemos y que ya hemos descrito en el capítulo 3. Recordar que contiene 60 imágenes presentando lesiones con regiones de velo azul-blanco, mayoritariamente casos de melanoma, y 30 imágenes que no contienen regiones de velo, donde 10 imágenes son diagnosticadas con clark nevus, y 20 con Dermal nevus.

En la parte de pre procesado habíamos extraído muestras de píxeles de entrenamiento y prueba tanto para regiones de velo como para no velo. Esos píxeles de entrenamiento hemos dicho que se constituyen de 150 muestra, 60 era de velo y 90 de no velo extraídas en cada una de las imágenes de nuestra base de datos. Gracias a estas muestras de entrenamiento podíamos extraer de ellas las características de color, para entrenarlas y obtener nuestro árbol de decisión.

En la fase de prueba, falta por probar este árbol y ver los resultados que no ofrece en sensibilidad y especificidad. Para ello en el pre procesado habíamos escogido de igual modo un total de 150 muestras de píxeles de test (o prueba) distribuidas de la misma forma que las muestras de entrenamiento (60 velo y 90 no velo), 80 de estas muestras eran las mismas que las de entrenamiento, y 70 eran totalmente distintas entre velo y no velo, escogidas de manera aleatoria según descrito el artículo base de este PFC. La razón por la cual no hemos usado totalmente las mismas muestras de entrenamiento es para realizar una prueba más correcta de nuestro árbol y obtener unos resultados más reales. Eso sí se han aprovechado unos mismo 80 recortes de entrenamiento, por la simple razón de reducir el tamaño de los datos.

Hemos probado estos píxeles de prueba tanto con nuestro árbol de decisión como con el del artículo *Automatic Detection of Blue-White Veil and Related Structures in Dermoscopy Images de M.Emre Celebi, Hitoshi Iyatomi, William V.Stoecker, Randy H.Moss, Harold S.Rabinovitz, Giuseppe Argenziano, H.Peter Soyer* [1], sabiendo que en el artículo se han basado en un set de imágenes distinto del nuestro y más amplio. Esta es la razón por la cual hemos obtenido un árbol de decisión distinto del árbol del artículo (en ilustración 45 se puede ver que el árbol del artículo es totalmente distinto del nuestro).

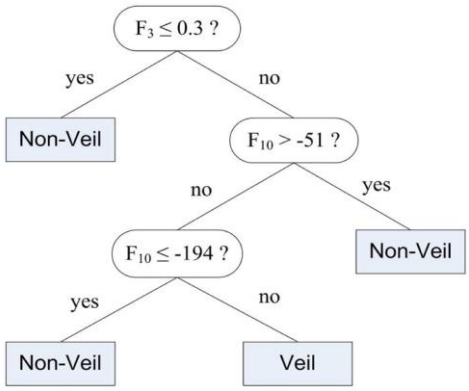


Ilustración 44. Árbol de decisión artículo

Vamos a calcular los porcentajes de sensibilidad (porcentaje de píxeles de velo detectados correctamente) y especificidad (porcentaje de píxeles de no velo detectados correctamente) y VPP (el valor predictivo positivo) siguiendo nuestro algoritmo pero también el del artículo. Para ello primero vamos a definir unos parámetros necesarios para realizar estos cálculos:

TP: Los recortes de píxeles verdaderos positivos → son los píxeles de velo que algoritmo dice que es velo

FN: Los recortes de píxeles falsos negativos→ son los píxeles de velo que algoritmo dice no velo

TN: Los recortes de píxeles verdaderos negativos→ son los píxeles de no velo que algoritmo dice no velo

FP: Los recortes de píxeles falsos positivos→ son los píxeles de no velo que algoritmo dice velo

La sensibilidad o fracción de verdaderos positivos se define como la probabilidad de clasificar correctamente a una muestra cuyo estado real sea el definido como positivo respecto la condición que estudia la prueba:

$$Sensibilidad = \frac{\mathrm{TP}}{TP + FN}$$

La especificidad se define como la probabilidad de clasificar correctamente a una muestra cuyo estado real sea el definido como negativo respecto la condición que estudia la prueba:

$$Especificidad = \frac{\mathrm{TN}}{TN + FP}$$

VPP o valor predictivo positivo no es un parámetro imprescindible en este cálculo, no obstante es un parámetro interesante de conocer, ya que tener una alta sensibilidad pero

bajo VPP nos lleva a un resultado muy poco útil realmente. Por lo tanto, una gran sensibilidad si no va acompañada de un elevado VPP de poco nos servirá en el día a día de la práctica médica:

$$Vpp = \frac{\text{TP}}{TP + FP}$$

La tabla 10 resume todos los cálculos de los recortes de píxeles de prueba (60 de velo y 90 de no velo) comparando con los dos árboles:

	Número recortes comparando con nuestro árbol	Número recortes comparando con árbol artículo
TP	58	24
TN	88	89
FP	2	1
FN	2	36

Tabla 10. Resumen número recortes de prueba

La siguiente tabla 11, nos presenta los cálculos de sensibilidad, especificidad, y Vpp a partir de los datos de la tabla 8:

	Árbol proyecto	Árbol artículo
Sensibilidad	96,67%	40%
Especificidad	97,78%	98,89%
VPP	96,67%	96%

Tabla 11. Resultados de sensibilidad, especificidad y Vpp

De la tabla 11 se ve que para nuestro árbol de decisión obtenemos unos resultados bastante buenos de sensibilidad y especificidad 96,67% y 97,78% respetivamente, acompañados de un elevado VPP. Cabe notar que si hubiéramos escogido unos recortes de test totalmente distintos que los de entrenamiento (recordar que 80 de los recortes de

prueba eran iguales que los de entrenamiento y los 70 restantes distintos) estos resultados lógicamente empeorarían ligeramente.

Sin embargo comparando nuestros recortes con el algoritmo del articulo base, obtenemos una sensibilidad baja de 40%, y para el caso de la especificidad incluso supera la obtenida con nuestro árbol, y un VPP casi igual para los dos árboles. Como ya se ha dicho, este resultado es debido a que nuestro set de imágenes es distinto del conjunto de imágenes usado en el artículo.

Lo que sí vale la pena comparar es nuestra Sensibilidad/Especificidad con las del artículo que han obtenido ellos con su set de imágenes y probando su árbol. En este caso y sabiendo que según el artículo han llegado a los siguientes resultados [1]:

Sensibilidad (artículo) = 84,33%

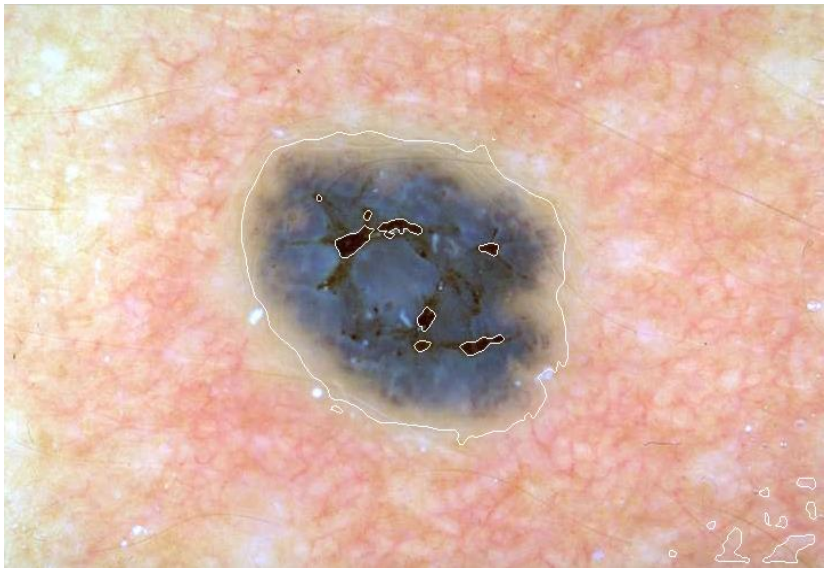
Especificidad (artículo) = 96,19%

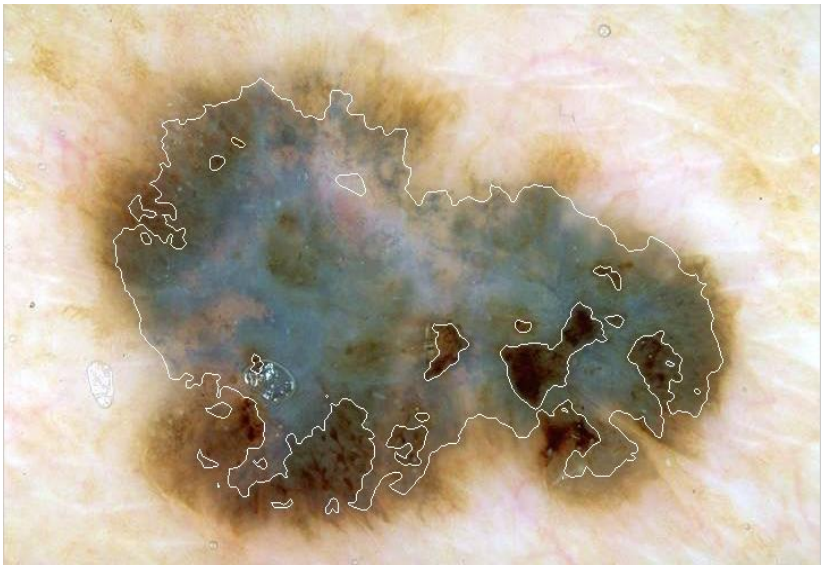
Deducimos que en cuyo caso nuestro proyecto supera sus resultados:

Sensibilidad (proyecto) = 96,67% > 84,33%

Especificidad (proyecto) = 97,78% > 96,19%

La ilustración 45 muestra los resultados de la detección de las regiones de velo para varias imágenes dermatoscópicas de nuestra base de datos. Se puede observar que el método presentado detecta la mayoría de las áreas de velo azul-blanco con una notable precisión.





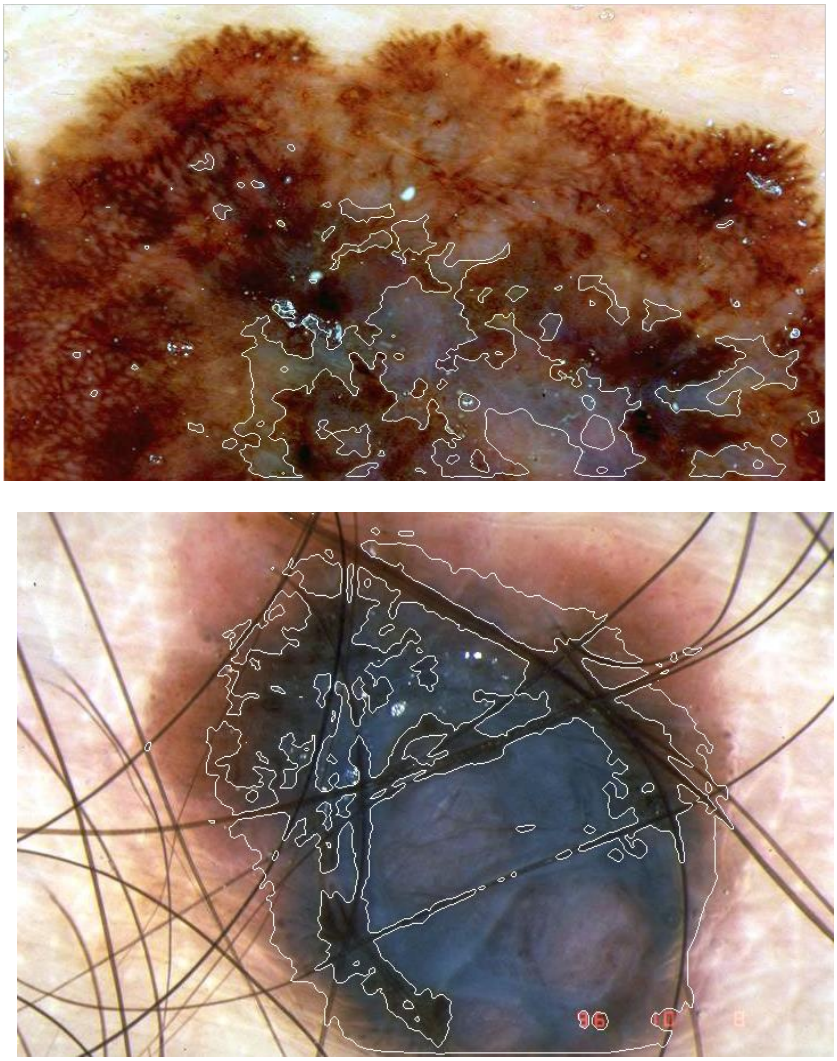


Ilustración 45. Ejemplos de resultados de detección de velo azul-blanco para 8 imágenes de nuestra base de datos. Los bordes del velo están delineados con líneas blancas.

CAPÍTULO 5: Conclusiones y líneas futuras

En este proyecto, se describe un método de aprendizaje automático para la detección de velo azul-blanco en imágenes dermatoscópicas. El método se compone de varias etapas que incluyen: pre-procesamiento, extracción de características, la inducción de un árbol de decisión aplicando el algoritmo C4.5, y finalmente después de un post-procesamiento, la aplicación de las reglas de decisión resultantes de dicho árbol.

Las zonas de velo con color blanco azulado detectadas, se caracterizaron utilizando unas características de color numéricas relativas según ha decidido nuestro algoritmo clasificador, que en conjunto arrojó una sensibilidad del 96,67% y una especificidad del 97,78% en un conjunto de 90 imágenes dermatoscópicas. Nuestras imágenes tienen una resolución de 768 × 512. El algoritmo detector del velo azul-blanco se ha realizado usando el potente software matemático Matlab versión 2013b sobre un ordenador Intel Core (TM) i7 CPU 1,60 GHz.

El objetivo establecido antes de la realización de este trabajo era el desarrollo de un sistema autónomo capaz de detectar regiones de velo azul-blanco en imágenes dermatoscópicas de lesiones pigmentadas de la piel, intentando mejorar los resultados obtenidos en el artículo base de este proyecto en cuanto a sensibilidad y especificidad. Una vez implementado el algoritmo y realizadas las pruebas pertinentes, se comprueba que el algoritmo supera efectivamente los resultados obtenidos en el artículo cumpliendo las siguientes finalidades:

- Es un sistema autónomo, pues los parámetros empleados son fijos para todas las imágenes, sin ninguna dependencia respecto a la imagen dermatoscópica de entrada.
- Consigue detectar el velo de la lesión presente en las imágenes, dando unos resultados precisos.

Por todo ello, se han alcanzado los objetivos propuestos inicialmente, quedando demostrado con los resultados obtenidos experimentalmente.

Cabe destacar la gran ayuda que nos ha dado el software Matlab. Gracias a sus numerosas librerías y herramientas, nos ha ofrecido un cómodo desarrollo de este algoritmo. Otra alternativa a Matlab hubiera sido el uso del software Octave, que a diferencia de Matlab se caracteriza por ser un programa gratuito. Sus herramientas en el campo de tratamiento digital de imágenes son parecidas a las de Matlab aunque ofrecen menos ayuda en la programación de algunas funciones, es decir que se necesita hacer varias funciones, que en Matlab vienen ya definidas en sus librerías. El desarrollo de este proyecto usando Octave nos hubiera costado un poco más de trabajo.

En cuanto a otras líneas futuras de trabajo, se podría ampliar el desarrollo de este proyecto para realizar un segundo clasificador que discrimina entre melanoma maligno y lesiones benignas en base de la presencia/ausencia del velo azul-blanco. Para ello tendríamos que tener ya diagnosticadas en nuestro conjunto de imágenes las lesiones malignas y las benignas por parte de un dermatólogo, para poder emplear las decisiones en el mismo algoritmo C4.5. Y así obtener un segundo árbol de decisión. Sin embargo

habría que emplear unas nuevas características a saber la circularidad y la elipticidad de los píxeles del velo detectado, características que se van a entrenar para el segundo clasificador, mediante el C4.5, y que nos clasificará dos decisiones: Melanoma maligno o lesión benigna.

También sabemos que el algoritmo presentado en este documento solo se centra en una de las características del melanoma en lesiones pigmentadas de la piel que es el velo azul-blanquecino. Otra línea futura interesante sería fusionarlo con otros algoritmos, por ejemplo que clasifican y analizan otras características del melanoma en el campo de la dermatoscopia, como los patrones globales o bien otras características de color que no se han tenido en cuenta para este trabajo, y así obtener un algoritmo muy potente capaz de ayudar al diagnóstico de esta enfermedad de la piel con más eficiencia.

5.1 Conclusión personal

El desarrollo de este proyecto me ha permitido adquirir más conocimientos y mejorar los que tenía sobre la herramienta de Matlab. Esta herramienta que he tenido la suerte de usarla durante toda mi carrera, y me he dado cuenta de lo potente que es después de la realización de este trabajo. El proyecto ha permitido también que me profundice más en el campo del tratamiento digital de imágenes, campo que conozco bastante ya que era mi intensificación en mi segundo ciclo de la carrera.

Además gracias a este proyecto, he tenido la oportunidad de ver la importancia del tratamiento digital de imágenes en el sector de la medicina en general y de la dermatología en particular.

Por último, aparte de servir para aprender y concienciar sobre el melanoma, ha sido un verdadero placer realizar un proyecto fin de carrera como este, por su gran aspecto humano. Ya que considero que su importancia reside en que con él se intenta ayudar al diagnóstico de una dura enfermad. Esta enfermedad se llama el cáncer y para muchos es el peor enemigo del ser humano.

CAPÍTULO 6: Referencias

[1] Automatic detection of blue-white veil and related structures in dermoscopy images de M.Emre Celebi, Hitoshi Iyatomi, **William V.Stoecker, Randy H.Moss, Harold S.Rabinovitz, Giuseppe Argenziano, H.Peter Soyer**.

[2] **SEOM**. [Online]
http://www.seom.org/es/informacion-sobre-el-cancer/info-tipos-cancer/melanoma?start=1

[3] **The Doctor's Doctor**. [Online]
http://www.thedoctorsdoctor.com/diseases/skin_pigmented_lesions.htm

[4] **Medscape**. [Online]
http://emedicine.medscape.com/article/1294801-overview

[5] **Servicio dermatología hospital del Mar**. [Online]
http://www.dermatologia.cat/es/unitatlesionspigmentadesimelanoma.html

[6] **MedlinePlus**. [Online]
https://www.nlm.nih.gov/medlineplus/spanish/ency/article/000850.html

[7] **Argenziano G, Soyer HP, De Giorgi V**, et al. Interactive Atlas of Dermoscopy. Milan, Italy: EDRA Medical Publishing & New Media; 2002.

[8] **Wikipedia**. [Online]
https://es.wikipedia.org/wiki/Dermatoscopia

[9] **Servicio dermatología hospital del Mar**. [Online]
http://www.dermatologia.cat/es/tecndermatoscopia.html

[10] **W. Stolz, A. Riemann, A. B. Cognetta, L. Pillet, W. Abmayr, D. Hölzel, P. Bilek, F. Nachbar, M. Landthaler, and O. Braun** – Falco, "ABCD rule of dermatoscopy: a new practical method for early recognition of malignant melanoma," European Journal Dermatology, vol. 4, pp. 521 – 527, 1994.

[11] **F. Nachbar, W. Stolz, T. Merkle, A. B. Cognetta, T. Vogt, M. Landthaler, P. Bilek, O. Braun – Falco, and G. Plewig**, "The ABCD rule of dermatoscopy: High prospective value in the diagnosis of doubtful melanocytic skin lesions," Journal of the American Academy of Dermatology, vol. 30, no. 4, pp. 551 – 559, April 1994.

[12] **G. Argenziano, G. Fabbrocini, P. Carli, V. De Giorgi, E. Sammarco, and M. Delfino**, "Epiluminescence microscopy for the diagnosis of doubtful melanocytic skin lesions," Archives of Dermatology, vol. 134, pp. 1563 – 1570, December 1998.

[13] **Argenziano G, Soyer HP, De Giorgi V**, et al. Interactive Atlas of Dermoscopy. Milan, Italy: EDRA Medical Publishing & New Media; 2002.

[14] **Hance GA, Umbaugh SE, Moss RH, Stoecker WV**. Unsupervised Color Image Segmentation with Application to Skin Tumor Borders. IEEE Engineering in Medicine and Biology. 1996; 15(1):104–111

[15] **Wikipedia**. [Online] http://es.wikipedia.org/wiki/Modelo_de_color_RGB#Modelo_de_color_RGB

[16] **Weiss GM, Provost FJ**. Learning When Training Data are Costly: the Effect of Class Distribution on Tree Induction. Journal of Artificial Intelligence Research. 2003; 19:315–354.

[17] **Gevers T, Smeulders AWM**. Color-Based Object Recognition. Pattern Recognition. 1999; 32(3):453–464.

[18] **Wikipedia**. [Online] https://es.wikipedia.org/wiki/Espacio_de_color_CIE_1931

[19] **Celebi ME, Kingravi HA, Uddin B**. A Methodological Approach to the Classification of Dermoscopy Images. Computerized Medical Imaging and Graphics. 2007; 31(6):362–373.

[20] **Oates T, Jensen D**. Large Datasets Lead to Overly Complex Models: An Explanation and a Solution. Proc of the 4th Int Conf on Knowledge Discovery and Data Mining. 1998:294–298.

[21] **Quinlan JR**. C4.5: Programs for Machine Learning. San Francisco, CA: Morgan Kaufmann; 1993.

[22] **Bruno López Takeyas**. [Online] http://www.itnuevolaredo.edu.mx/takeyas/Apuntes/Inteligencia%20Artificial/Apuntes/tareas_alumnos/C4.5/C4.5(2005-II-B).pdf

[23] **Pratt WK**. Digital Image Processing: PIKS Inside. New York, NY: John Wiley & Sons; 2001.