Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Implementación y evaluación de técnicas de mejoras de la imagen

Autor: Juan Carlos Hidalgo Palomo
Tutores: Carmen Serrano Gotarredona
Begoña Acha Piñero

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

Sevilla, 2023

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Implementación y evaluación de técnicas de mejoras de la imagen

Autor:
Juan Carlos Hidalgo Palomo

Tutores:
Carmen Serrano Gotarredona
Begoña Acha Piñero

Dpto. de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Trabajo Fin de Grado: Implementación y evaluación de técnicas de mejoras de la imagen

Autor: Juan Carlos Hidalgo Palomo
Tutores: Carmen Serrano Gotarredona
Begoña Acha Piñero

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2023

El Secretario del Tribunal

A mi familia
A mis maestros

Resumen

En este proyecto se definen, en primer lugar, una serie de medidas estadísticas para la calidad de la imagen digital, denominadas referenciadas y no referenciadas, diferenciándose, fundamentalmente, en la comparación con la existencia de la imagen original o no.

A través de estas medidas, se describen y aplican unos métodos, algunos sirven para eliminar imperfecciones de la imagen, o como se conoce en el ámbito del tratamiento digital de imágenes, el ruido, otros sirven para mejorar o visualizar mejor otras características de la imagen, como la croma, la intensidad, la nitidez de los bordes, etc.

Finalmente, se implementaran y mostraran los resultados del tratamiento digital de las imágenes realizados en el software MATLAB.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Notación y Abreviaturas

ISIC	International Skin Imaging Collaboration
USM	Unsharp masking
HBF	High Boost Filtering
MSE	Minimum square error
PSNR	Peak Signal-to-Noise Ratio
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
RAE	Real Academia Española
A^*	Conjugado
:	Tal que
<	Menor o igual
>	Mayor o igual
\	Backslash
⇔	Si y sólo si

1 INTRODUCCIÓN

El objetivo técnico de este proyecto es evaluar los principales métodos o técnicas básicas de mejora y eliminación de ruido para una serie de imágenes médicas obtenidas de la base de datos pública *International Skin Imaging Collaboration* (ISIC).

Para ello se emplearán diferentes medidas de calidad de la imagen, clasificadas en referenciadas y no referenciadas, distinguiendo un tipo del otro la existencia o no de la imagen original para su comparación.

A las medidas referenciadas también se las conoce como medidas de fidelidad, y las mejoras o cambios que se le apliquen a la imagen están relacionados con la imagen original.

Por otro lado, las medidas no referenciadas comparan propiedades y atributos de la propia imagen.

Describiremos brevemente, qué es el ruido digital en la imagen, cómo afecta y qué principales técnicas usaremos para eliminarlo todo lo que sea posible.

Finalmente, implementáramos los métodos de eliminación de ruido y de mejora de la imagen para ver los resultados de las medidas referenciadas y no referenciadas, a través de la aplicación MATLAB discutiendo los resultados.

2 MEDIDAS DE CALIDAD DE LA IMAGEN

2.1 Referenciadas

En primer lugar, empezaremos definiendo qué son las medidas de calidad referenciadas, también conocidas como medidas de fidelidad. En este sentido, la fidelidad se conoce como una medida de referencia completa de la calidad de la imagen, ya que se requiere la referencia o la imagen original sin distorsiones para estimar dicha fidelidad. Por tanto, es solo un indicador de la calidad general de una imagen [1-2].

Las principales medidas de fidelidad que vamos a analizar en este proyecto son:

2.1.1 El error cuadrático medio, en inglés *Mean Square Error* (MSE)

Esta medida es muy usada en la comparación de datos de forma general, mide el promedio de los errores al cuadrado, compara un valor predicho y un valor observado o conocido. En el caso de las imágenes vamos a comparar la imagen original, **f**, con la imagen a tratar **g.**
En primer lugar, del espacio de color RGB, el cual está formado por 3 planos de cada color primario, R, G y B, vamos a pasar al espacio de color CIE L*a*b*, ya que éste permite separar la luminosidad o brillo (plano L) de las coordenadas cromáticas [a*= coordenadas rojo/verde (+a indica rojo, -a indica verde); b* = coordenadas amarillo/azul (+b indica amarillo, -b indica azul)].

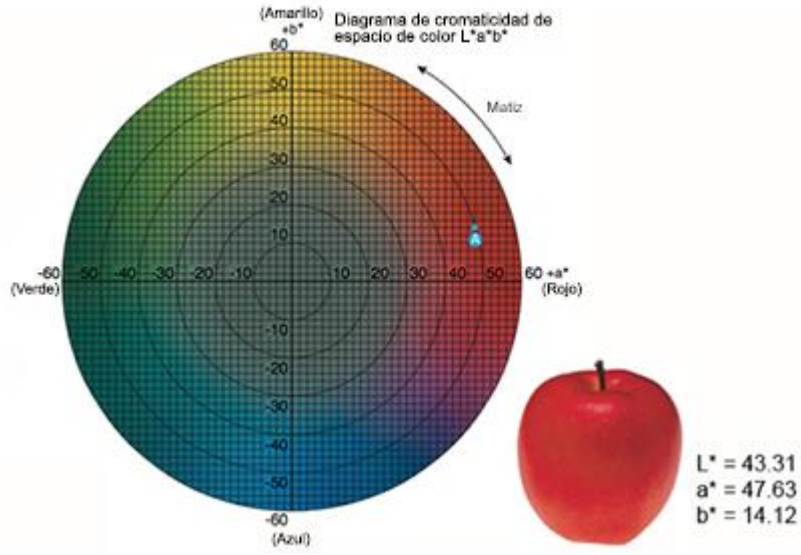


Figura 1. Espacio de color CIE L*a*b*.

El MSE para las imágenes se define como:

$$MSE_{CIELAB} = \frac{1}{N} \sum_{1 \le n \le N} \|f_n - g_n\|^2 \qquad (2-1)$$

donde N es el número de píxeles de la imagen, $\| \|$ representa la distancia Euclídea, f_n y g_n hace referencia a los píxeles en las imágenes en ese espacio de color acotado de $1 \le n \le N$.

2.1.2 La proporción máxima o pico de la relación señal a ruido, del inglés *Peak Signal-to-Noise Ratio*, (PSNR)

De forma general, este concepto mide la relación entre la máxima energía de una señal y el ruido que afecta a la misma. Debido a que muchas señales tienen un gran rango dinámico, el PSNR se expresa generalmente en escala logarítmica, utilizando como unidad el decibelio.

Pero uno de sus usos principales es como medida de calidad y fidelidad en el ámbito del análisis, reconstrucción, compresión y estudio de las imágenes.

Por tanto, para las imágenes se define como la relación entre los valores o píxeles máximos del rango y el error cuadrático medio (MSE).

$$PSNR_{CIELAB} = 10 \log_{10} \left\{ \frac{255^2}{\frac{1}{N} \sum_{1 \leq n \leq N} \|f_n - g_n\|^2} \right\} \quad (2-2)$$

$$PSNR_{CIELAB} = 10 \log_{10} \left\{ \frac{255^2}{MSE_{CIELAB}} \right\} \quad (2-3)$$

donde el rango dinámico de las componentes de color es de $[0,255]$.

El error cuadrático medio guarda una relación inversamente proporcional al pico de la relación señal a ruido, es decir que, si para un mismo escenario, la primera aumenta, la segunda disminuye y viceversa.

En lo que se refiere a la codificación para la compresión, el PSNR es mayor cuanto mejor es la codificación. Además el comité MPEG (*Moving Picture Experts Group*) definido como un grupo de trabajo de expertos que se formó por la ISO (Organización Internacional de Normalización) y la IEC (Comisión Electrotécnica Internacional) para establecer estándares para el audio y la transmisión de video, emplea un valor umbral informal de 0,5 dB en el incremento del PSNR para decidir si se incluye una determinada mejora en un algoritmo de codificación, ya que se considera que este aumento del PSNR es apreciable visualmente [3].

Estas medidas las hemos tomado en el plano L del espacio de color CIE Lab, ya que el rango está acotado a un solo plano, pero también las hemos obtenido del espacio de color RGB, con la única diferencia que el valor del MSE es la media aritmética de los tres planos de color R, G y B.

2.2 No referenciadas

Se trata de medidas realizadas en una sola imagen, sin disponer de la imagen original con la que comparar, por tanto, todas las propiedades, atributos, transformaciones, etc., se realizan dentro de la misma imagen.
En un estudio llevado a cabo por Sabakis AE, Stephen PE y Loui AC, llamado "*Evaluation of image appeal in consumer photography*", se elaboró un ranking de los atributos de imagen más observados por los humanos y entre los más votados están tres medidas no referenciadas, como son: el **contraste**, el colorido (***colorfulness***) y la nitidez de los bordes de la imagen (***sharpness***).

2.2.1 El contraste

Este término tiene varios significados, aunque de forma general la Real Academia Española (RAE) lo define como la oposición, contraposición o diferencia notable que existe entre personas o cosas. El contraste se usa en el diseño gráfico para diferenciar elementos a nivel de color, por forma o por disposición de los mismos.
A nivel de color, podemos diferenciar siete tipos para contrastar: de color puro, de claro a oscuro, de temperatura, de colores complementarios, de saturación, de cantidad y simultáneo.
Aunque en este proyecto nos vamos a centrar en la definición relacionada con el tratamiento de imágenes que, según la RAE, es la relación existente entre el brillo, luminancia o intensidad de cada uno de los puntos que forman la imagen.

De forma general se representa como:

$$\frac{Diferencia\ de\ Luminancia}{Luminancia\ Promedio} \tag{2-4}$$

Y podemos medirlo gracias a la fórmula del contraste de Michelson [4]:

$$\frac{I_{\max} - I_{min}}{I_{\max} + I_{min}} \tag{2-5}$$

donde I_{max} es la intensidad o luminancia máxima e I_{min} es la intensidad mínima.

Y a la fórmula del contraste de Weber:

$$\frac{I - I_b}{I_b} \tag{2-6}$$

desarrollada por los científicos Ernst Heinrich Weber (1795–1878) y Gustav Theodor Fechner (1801–1887) en la ley de Weber-Fechner. Donde I es la luminancia de la características e I_b la luminancia del fondo. Usada en los casos donde existen una proporción pequeña de características dentro de un gran fondo uniforme, o lo que es lo mismo, la luminancia promedio es muy similar a la luminancia de fondo [5].
Esta última formula también aparece en muchos documentos de forma abreviada como:

$$\frac{\Delta L}{L} \tag{2-7}$$

De esta forma, vamos a calcular el contraste localizado en un área limitada, realizando la diferencia entre su valor máximo y su valor mínimo.

2.2.2 El colorido (*colorfulness*)

Se define como la presencia y la viveza del color en toda la imagen, da una sensación visual según la cual el color percibido de un área parece ser más o menos cromático.
Según este concepto, cada color tiene dos atributos perceptivos [6]:

1. La saturación: es el colorido de un área o de toda la imagen con relación a su propio brillo.
2. La croma: es el colorido de un área relativo al brillo de otro color que aparece en blanco bajo condiciones similares de visualización.

Podemos ver también algunos de estos atributos representados en el espacio de color CIE L*C*h, similar al espacio CIE L*a*b*, que se sirven del mismo diagrama, pero el primero usa coordenadas cilíndricas, mientras que el segundo coordenadas rectangulares.
L (Lightness) es la luminosidad, C (Chroma) es la croma y h (Hue) es el matiz.

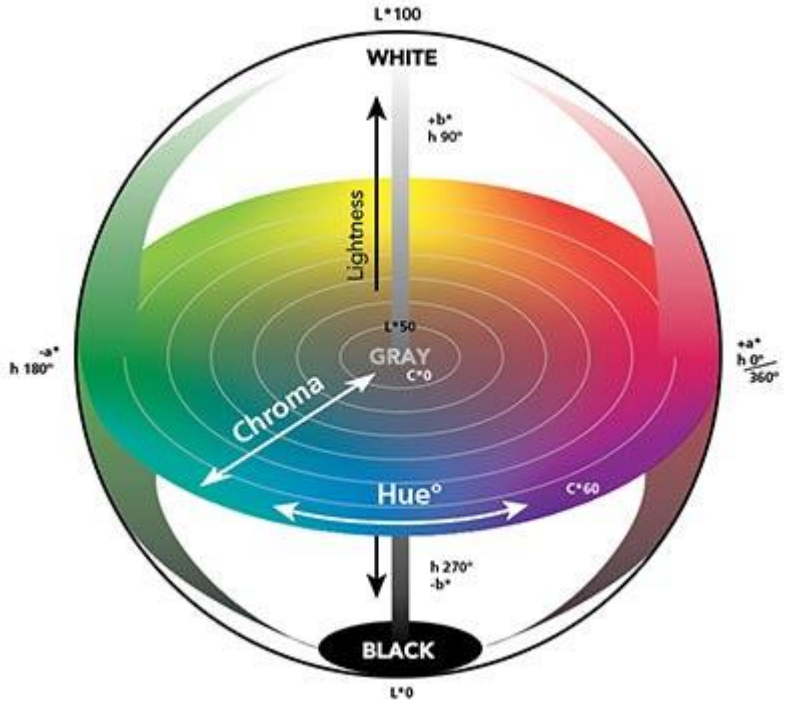


Figura 2. Espacio de color CIE L*C*h.

Para calcular el colorido de una imagen sumamos la media de todos los valores de la imagen más la variación estándar de la misma, la cual indica qué tan dispersos están los datos con respecto a la media.

2.2.3 La nitidez de la imagen (*sharpness*)

De forma general, la nitidez es la claridad con la que se perciben los detalles. En las imágenes se suele medir el gradiente entre píxeles.

El gradiente, en el caso de una imagen en dos dimensiones (caso discreto), se basa en las diferencias entre los niveles de grises de la imagen adyacentes en la misma fila y columna [7]:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} \qquad (2-8)$$

Para su cálculo, realizaremos el promedio del módulo del gradiente.
La nitidez especifica como los bordes de cada elemento existente en la imagen no se ven borrosos.

3 MÉTODOS DE ELIMINACIÓN DE RUIDO Y MEJORA EN IMÁGENES DIGITALES

3.1 El ruido en imágenes

El ruido en las imágenes se define como una variación del color y la intensidad de manera aleatoria de algunos píxeles de la imagen, provoca una mala visualización de la calidad y resolución de las mismas, éste se debe a múltiples factores como: ruido de Poison debido a la naturaleza corpuscular de la luz, ruido térmico de los receptores (ejemplo: corriente de oscuridad en la CCD), ruido de cuantificación, ruido de compresión con pérdidas, etc. Surgen principalmente, en la fase de toma o adquisición y transmisión de la imagen [8].

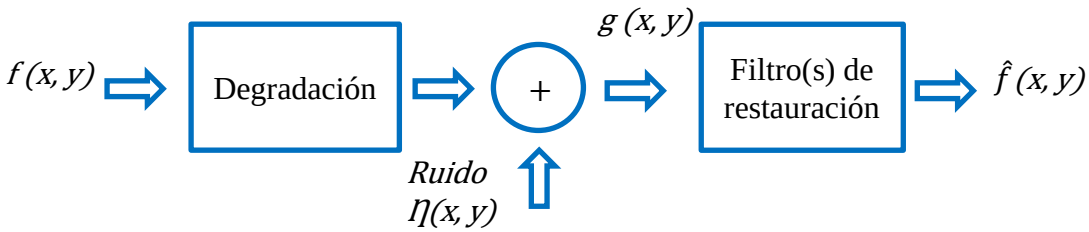


Figura 3. Modelo del proceso de degradación/restauración de una imagen.

Podemos modelarlos con dos tipos de ruidos principalmente, llamados:

3.1.1 Ruido Gaussiano

Está asociado con la radiación electromagnética, muestra una densidad de probabilidad que responde a una distribución normal o distribución de Gauss, que siguen todos los píxeles de la imagen afectada.

La función densidad de probabilidad (**FDP**), se define como la probabilidad relativa de que la variable aleatoria que tiene asociada tome un determinado valor [8], esta probabilidad viene dada por la formula:

$$P[a \leq X \leq b] = \int_a^b f_X(x)\, dx \qquad (3-1)$$

La FDP de Gauss se representa con la expresión:

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-\frac{(z-\bar{z})}{2\sigma^2}} \qquad -\infty < z < \infty \qquad (3-2)$$

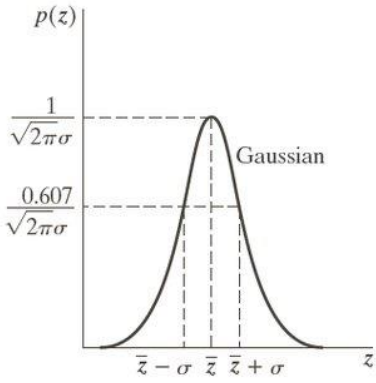


Figura 4. FDP Ruido gaussiano.

Donde z representa la intensidad, $\bar{z}$ es la media o **valor esperado** de la intensidad y σ es la **desviación estándar**.
Estos dos últimos conceptos son muy importantes en el tratamiento y análisis del ruido, ya que aportan información de cómo están distribuidos los píxeles. También se usa el parámetro denominado **varianza**, que no es más que **desviación estándar** al cuadrado, σ^2.

Representación del ruido gaussiano en la Figura 5:

Figura 5. Ruido Gaussiano.

3.1.2 Ruido Impulsivo (sal y pimienta)

El valor del píxel no tiene relación con el valor original, sino que toma valores alrededor del máximo o alrededor del mínimo, en el caso de escala de grises, blanco (de ahí que se le llame sal) o negro (de ahí que se le llame pimienta), respectivamente. Se modela con una función de distribución no gaussiana o escalón. Vemos en la Figura 7 el efecto del ruido, donde algunos píxeles aleatoriamente toman unos valores y otros toman otros.

Su función densidad de probabilidad (FDP) [8] se define como:

$$p(z) = \begin{cases} P_s & z = 2^k - 1 \\ P_p & z = 0 \\ 1 - (P_s + P_p) & z = V \end{cases} \qquad (3-3)$$

donde el rango de intensidades, z, es $[0, 2^k - 1]$ y V es cualquier valor entero entre $0 < V < 2^k - 1$.

Cuya representación gráfica se ve en la Figura 6:

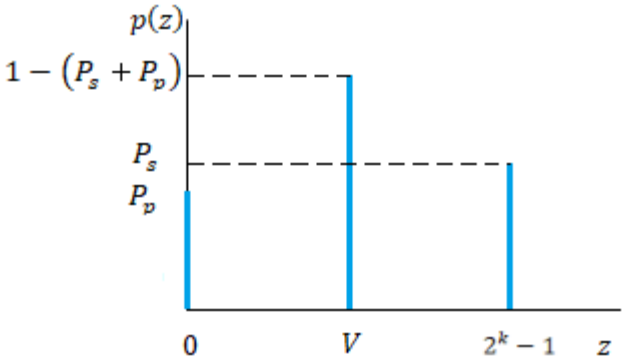


Figura 6. FDP Ruido sal y pimienta.

A una imagen que ha sido degrada con ruido sal y pimienta, cuyos valores de intensidad sigue la PDF anterior, se le asigna un 0 (pimienta: negro) al ruido con una probabilidad P_p. De igual forma, el ruido toma un valor $2^k - 1$ (el valor máxino de intensidad, sal: blanco) con una probabilidad P_s. El resto valores permanecen igual con lo que resta de probabilidad, $1 - (P_s + P_p)$ [8].

La probabilidad P, es la probabilidad con la que un píxel es corrompido por el ruido de sal y pimienta, $P = P_s + P_p$.

A P tambien se le conoce como densidad de ruido.

Representación del ruido sal y pimienta en una imagen, Figura 7:

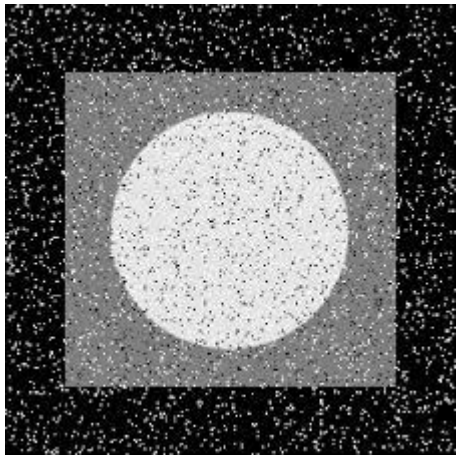


Figura 7. Ruido Impulsivo.

3.2 Métodos de eliminación del ruido

Existen métodos para solventar este problema y eliminarlo en gran parte, algunos funcionan mejor dependiendo del tipo y características del ruido. Nos vamos a centrar en dos de los principales.

3.2.1 Filtro de media:

Para este método, consideremos un píxel rodeado por un grupo pequeño de píxeles dentro de la imagen, este grupo se denomina **vecindario.**
Para poder aplicarlo es necesario los siguientes requisitos:

- La imagen es relativamente constante en un vecindario, no hay apenas variación.
- Las variaciones en los píxeles son debidas al ruido.

Algunos de los vecindarios o agrupaciones de píxeles más utilizados son: un cuadrado 3x3 (donde el píxel está conectado con 8 vecinos), una ventana de 5 píxeles, donde el píxel central está conectado con 4 o un grupo de barras, donde el píxel solo está conectado arriba y abajo o a la derecha e izquierda [9].

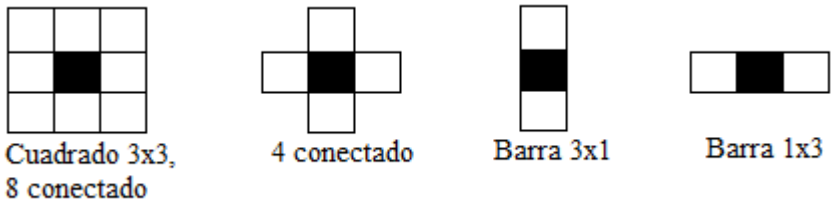


Figura 8. Ejemplos de vecindarios.

Para un vecindario 3x3, 8-conectado:

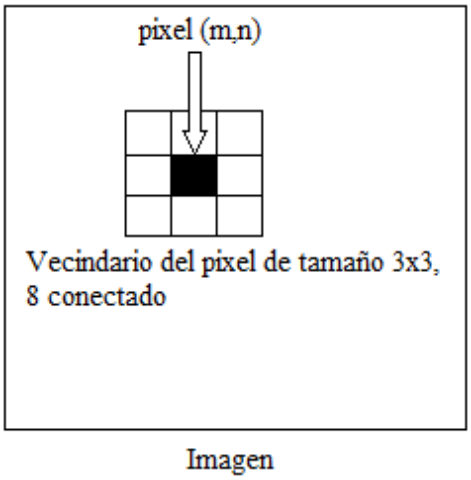


Figura 9. Vecindario 3x3, 8 conectado de un píxel.

$$g(m,n) = \frac{1}{9} \sum_{p=-1}^{1} \sum_{q=-1}^{1} f(m+p, n+q) \qquad (3-4)$$

donde f(m,n) es la imagen de entrada u original, p y q son las posiciones alrededor del píxel y g(m,n) es la imagen de salida tras el filtrado.

$$g(m,n) = \frac{1}{9}\,[f(m-1,n-1) + f(m-1,n) + f(m-1\,,n+1) + f(m,n-1) + f(m,n) + f(m,n+1) + f(m+1,n-1) + f(m+1,n) + f(m+1,n+1)] \qquad (3\text{–}5)$$

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Representación del resultado del filtro de media 3x3 tras aplicársele a la primera imagen con ruido gaussiano, Figura 10:

Figura 10. A la izquierda, imagen con ruido gaussiano; a la derecha , imagen tras aplicarle el filtro de media 3x3.

Se aprecia claramente, en la Figura 10, el suavizado de la imagen tras aplicar el filtro de media. Hemos usado hasta ahora este método para la intensidad de la imagen, pero es extensible a las imágenes en color también, únicamente se aplicaría el filtro a cada una de las componentes del espacio de color.

A continuación, mostramos en la Tabla 1 las ventajas y desventajas de este método:

Tabla 1. Ventajas y desventajas del filtro de media.

Ventajas	Desventajas
Elimina bien el ruido gaussiano y uniformemente distribuido en áreas relativamente homogéneas de la imagen	Emborrona los bordes de los elementos de la imagen.
Es el más sencillo de implementar.	No es muy efectivo contra el ruido impulsivo.

3.2.2 Filtro de mediana:

La mediana de un grupo de muestras, que en nuestro caso se trata de un vecindario, es el valor que divide al grupo justo por la mitad, es decir, una mitad de píxeles estarán por debajo de la mediana y los píxeles restantes estarán por encima de la mediana [9].

Por ello, el grupo de píxeles generalmente es un número impar, si no se realizan aproximaciones. Acudimos a un ejemplo para ilustrarlo:

En un vecindario con un número impar de píxeles como:

$$\begin{bmatrix} 100 & 44 & 80 \\ 77 & 112 & 10 \\ 82 & 144 & 22 \end{bmatrix} \quad (3-6)$$

Realizamos ahora el ordenamiento de los píxeles de menor a mayor para obtener la mediana, que sería el píxel situado justo en el medio:

$$[10\ 22\ 44\ 77\ \mathbf{80}\ 82\ 100\ 112\ 144] \quad (3-7)$$

El filtro de mediana sustituye ahora la mediana, en este ejemplo sería 80, en el píxel central del vecindario:

$$\begin{bmatrix} 100 & 44 & 80 \\ 77 & \mathbf{80} & 10 \\ 82 & 144 & 22 \end{bmatrix} \quad (3-8)$$

Representación, en la Figura 11, del resultado del filtro de mediana 3x3 tras aplicársele a la primera imagen con ruido sal y pimienta:

Figura 11. A la izquierda, imagen con ruido sal y pimienta; a la derecha, imagen tras aplicarle el filtro de mediana 3x3.

Vemos cómo se comporta este filtro para este tipo de ruido, prácticamente lo elimina por completo.

Al igual que el método anterior, mostramos en la Tabla 2, las ventajas y desventajas de este filtrado:

Tabla 2. Ventajas y desventajas del filtro de mediana.

Ventajas	Desventajas
Elimina bien el ruido impulsivo o con función de densidad de probabilidad similar.	Puede redondear esquinas o distorsionar formas si los elementos de la imagen tienen bordes pronunciados.
No emborrona los bordes.	Si el soporte es demasiado grande, elimina detalles.

3.3 Métodos de mejora de la imagen

En el proceso digital del tratamiento de imágenes, uno de los puntos principales es la mejora y realce de las imágenes para una mejor visualización de las mismas o para recuperar o restaurar imágenes distorsionadas o degradadas.
A pesar de que hoy día los dispositivos que toman las imágenes disponen de una muy buena calidad técnica, un gran rango de resolución, buen enfoque, reducción de vibraciones, etc., en el proceso de adquisición de la misma, es necesario en muchos casos que se mejoren o realcen características de la imagen tomada para una mayor visualización o mejora de la calidad.
En la mayoría de los casos, la mejora buscada en una imagen está enfocada a lograr una o más de las siguientes características deseadas:

I. Representación limpia y clara de los objetos de la imagen, sin ruido ni imperfecciones.
II. Buen contraste y visibilidad de los detalles.
III. Brillo uniforme o equilibrado en toda la imagen, lo que puede requerir que las áreas oscuras se aclaren y las áreas con brillo excesivo se atenúen.
IV. Reproducción fiel de tonos matices de color, con especial atención al tono de piel y el matiz en imágenes de humanos.
V. Buen balance de color para dar como resultado una apariencia agradable.
VI. Bordes nítidos y bien definidos de objetos o regiones en las imágenes.

Las imágenes en color se pueden procesar para mejorarlas tratando cada componente de manera individual y con ciertas condiciones, teniendo especial cuidado en el espacio de color RGB, ya que puedo haber cambios en cuanto al tono y la saturación. Por eso se suele usar también otros espacios de color que separe esas características, como el HSV o HSI [10].

Existen diferentes técnicas o métodos para la mejora o realce de las imágenes, dependiendo de la característica o características que se quieran mejorar, vamos a tratar en este proyecto los siguientes:

a. Ecualización del histograma

Se usa para mejorar el contraste de la imagen a través de su histograma, volviendo más uniforme.

b. Enmascaramiento del desenfoque

Más conocido por su traducción en inglés, *unsharp masking* (USM). Es una técnica de mejora de la nitidez de la imagen a través de la aplicación de un filtro con una máscara concreta.

c. Filtro de refuerzo de frecuencias altas

Del inglés, *high boost filter* (HBF). También es una técnica para la mejora de la nitidez de la imagen, funciona bien en áreas de gran detalle, pero no tan bien en áreas planas o suaves.

3.3.1 Ecualización del histograma

En primer lugar, vamos a definir qué es el histograma en una imagen digital, éste es la representación gráfica, generalmente en forma de barras, del número de píxeles que corresponden a cada intensidad o tonalidad y en cada canal del espacio de color.

Es una característica que ofrece un valor estadístico. Su análisis permite comparar contrastes e intensidades entre imágenes.

Puede ser modificado para ver cómo afectaría a la imagen y producir así cambios en ella.

Un ejemplo de su uso es separar en blanco y negro una imagen digital sin que se pierdan las propiedades fundamentales de la misma, para ello hay que elegir un valor umbral que delimite la separación, ese valor se situará donde el histograma de la imagen forme un valle. Este ejemplo se llama binarizar una imagen, se puede apreciar ese efecto en la Figura 12.

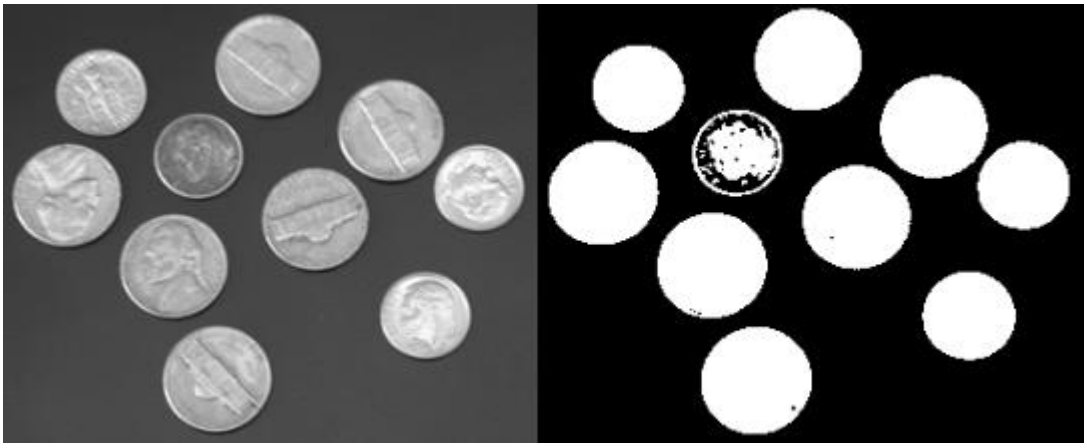


Figura 12. Binarización del histograma.

En el campo de la medicina, por ejemplo, para las imágenes radiológicas, es muy útil para separar huesos de tejidos u órganos.

En el capítulo 4 mostraremos cómo se ha implementado la binarización del histograma en MATLAB.

Tras ver qué es un histograma y cómo se comporta, vamos ahora la técnica en cuestión para la mejora de la imagen, denominada **ecualización del histograma**.

Esta técnica mejora el contraste de la imagen repartiendo, más o menos, uniformemente los valores en el histograma.

Matemáticamente, se pretende encontrar una transformación para lograr un histograma con una distribución uniforme.

Es una transformación de intensidad y se define [11], en una función continua, como:

$$s = T(r) \quad 0 \leq r \leq L - 1 \tag{3 – 9}$$

Y la trasformada inversa:

$$r = T^{-1}(s) \quad 0 \leq s \leq L - 1 \tag{3 – 10}$$

donde ***r*** indica la intensidad de la imagen, en rango de valores [0, L-1], representando 0 como negro y L-1 como blanco. Y ***s*** son los valores de intensidad de salida, tras la trasformación.

Aproximando el histograma de la imagen por su función densidad de probabilidad, p_r, y aplicando la transformación $s = T(r)$, obtendremos la imagen ecualizada.

La función $T(r)$ surge del siguiente desarrollo, (suponemos un histograma normalizado entre 0 y 1):

$$\int_0^1 p_r(r)\, dr = 1 \tag{3 – 11}$$

Ahora tenemos en cuenta, también, la función densidad de probabilidad de la imagen transformada, p_s, la cual queremos que sea uniforme.

$$\int_0^1 p_s(s)\, ds = 1 \tag{3 – 12}$$

Cuyas representaciones se puede ver en la Figura 13:

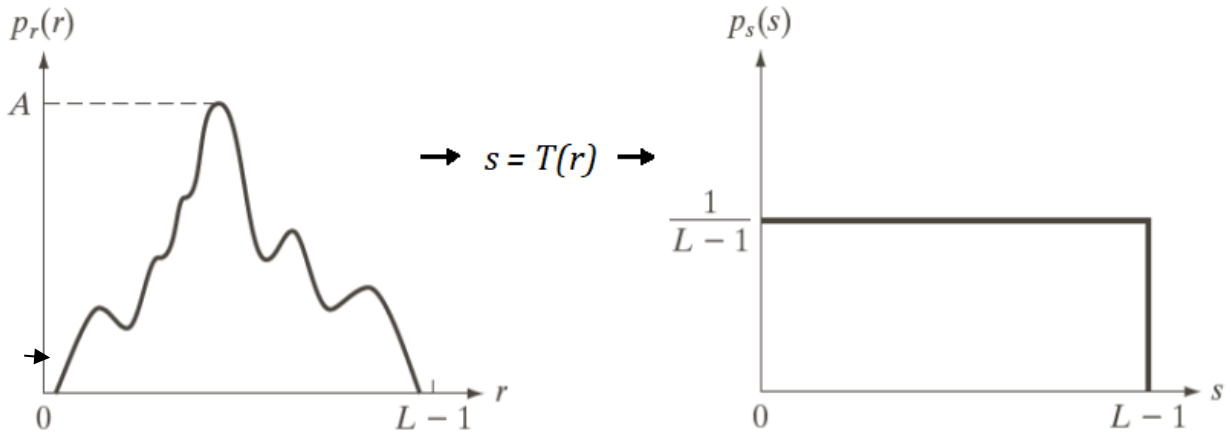


Figura 13. A la izquierda, FDP aleatoria de una imagen; a la derecha, FDP uniforme, resultado de la transformación.

Por tanto:

$$\int_0^1 p_r(r)\, dr = \int_0^1 p_s(s)\, ds \tag{3 – 13}$$

$$p_s(s)\, ds = p_r(r)\, dr \,|_{r = T^{-1}(s)} \tag{3 – 14}$$

$$p_s(s) = p_r(r) \frac{dr}{ds} |_{r = T^{-1}(s)} = 1 \qquad (3-15)$$

Finalmente, la transformación queda como:

$$s = T(r) = (L-1) \int_0^r p_r(w)\, dw \qquad (3-16)$$

Este desarrollo se ha realizado para una función continua, pero la imagen digital se representa por valores discreto, por tanto, para una imagen de dimensiones M x N con L niveles discretos, su función densidad de probabilidad es:

$$p_r(r_k) = \frac{n_k}{MN} \qquad k = 0,1,2,\dots,L-1 \qquad (3-16)$$

donde r_k es el valor de la intensidad del píxel en la posición k, y n_k es el número de píxeles existentes en el valor r_k correspondiente.

Aplicamos ahora la transformación discreta, quedando la fórmula:

$$s_k = T(r_k) = (L-1) \sum_{j=0}^{k} p_r(r_j) = \frac{L-1}{MN} \sum_{j=0}^{k} n_j \qquad k = 0,1,2,\dots,L-1 \qquad (3-17)$$

3.3.2 Enmascaramiento del desenfoque

Originalmente este método surgió en la fotografía analógica, también llamada de cuarto oscuro, la cual necesitaba de un proceso químico para capturar la imagen en papel, película, u otro material. El procedimiento consistía, de manera resumida, en copiar por contacto el negativo de la imagen original en una placa de vidrio de bajo contraste, esta impresión después se volvía a colocar por contacto en el negativo, pero por la otra cara, de tal forma que cuando pasaba la luz a través del negativo se observaba la imagen más nítida, ya que solo se cancela la información de bajas frecuencias [12].

El enmascaramiento del desenfoque, más conocido comúnmente por si traducción en inglés, *unsharp masking* (USM) es una técnica cuyo principal objetivo es mejorar la nitidez de las imágenes digitales.

Su nombre se debe a que el proceso usa una imagen borrosa o poco nítida, f_b, para crear una máscara, m, que solo posee información de los bordes de la imagen original, f. Esta máscara, llamada máscara de enfoque, se combina con la imagen original, creando una nueva, g, menos borrosa que la tomada inicialmente [13]. De tal forma que:

$$m(x,y) = f(x,y) - f_b(x,y) \qquad (3-18)$$

$$g(x,y) = f(x,y) + k * m(x,y) \qquad (3-19)$$

Tomando el promedio ponderado de la imagen original y borrosa, podemos escribir la fórmula:

$$g(x,y) = (k+1) * f(x,y) - k * f_b(x,y) \qquad (3-20)$$

Donde k especifica qué parte de la máscara es añadida, según el valor que tome:

- Para $\boldsymbol{k=1}$ se conoce como **enmascaramiento del desenfoque (USM)**.
- Para $\boldsymbol{k>1}$ se trata del filtrado de refuerzo de frecuencias altas, debido a que se aumentan aún más las componentes de alta frecuencia, ya que se le está dando más peso a la imagen enmascarada.

La máscara de enfoque no hace más que, en término de procesamiento de señales, amplificar las componentes de alta frecuencia.

En ocasiones, aunque se mejora la nitidez de la imagen, puede suceder que se distorsionen elementos de la imagen.

El enmascaramiento del desenfoque se puede controlar a través de tres parámetros:

I. La cantidad, que controla la fuerza del efecto de la nitidez, se mide en porcentaje. A mayor porcentaje de cantidad, mejor será el contraste de los píxeles nítidos.
II. El radio, el cual decide el área o la región alrededor de los bordes que se va a tratar.
III. El umbral (*threshold*), que elige la región o regiones en la que se mejore la nitidez a partir de ese valor umbral.

[14] Una popular máscara 3x3 usada en esta técnica es:

$$\begin{bmatrix} -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \\ -\frac{1}{8} & 2 & -\frac{1}{8} \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \end{bmatrix} \qquad (3-21)$$

3.3.3 Filtro de refuerzo de frecuencias altas

Los filtros de altas frecuencia, debido a su naturaleza, son aquellos que refuerzan, tal y como su nombre dice, las altas frecuencias y eliminan o dan menos peso a las frecuencias bajas de la imagen.

A nivel visual, resaltan los cambios bruscos de intensidad de la imagen digital, en cambio, reducen, minimizan o incluso eliminan en algunos casos los cambios suaves o degradados. Por tanto, son muy útiles para mejorar, resaltar y separar los bordes de las imágenes.

Existen numerosos filtros de alta frecuencia, pero nos vamos a centrar en el filtro de refuerzo de frecuencias altas, más conocido popularmente por su traducción en inglés, ***High Boost Filter* (HBF).**

Este filtro al ser de frecuencias altas enfatiza los valores de los píxeles de dichas frecuencias sin eliminar los componentes de bajas frecuencias, solo las suaviza. Por lo que son muy útiles en imágenes oscuras.

Agudiza los bordes de la imagen gracias a la amplificación de un factor que posee, obteniéndose una imagen más clara.

Por lo que se logra una mejora de la nitidez en la imagen digital, sobre todo, en áreas de alto detalle, pero poco en áreas planas o suaves.

Este filtrado se puede expresar con la siguiente formula [13]:

$$f_{HB}(m,n) = A * f(m,n) - f_{LPF}(m,n) \qquad (3-22)$$

donde $f(m,n)$ es la imagen original, A se denomina factor de amplificación que controla los pesos, $f_{LPF}(m,n)$ es la imagen filtrada de paso bajo e $f_{HB}(m,n)$ es finalmente la imagen filtrada con HBF.

Además, el valor del factor de amplificación tiene que ser: $A \geq 1$

Si ahora sumamos y restamos $f(m,n)$, la formula queda:

$$f_{HB}(m,n) = (A-1) * f(m,n) + f(m,n) - f_{LPF}(m,n) \qquad (3-23)$$

Si a la imagen original le eliminamos las componentes de baja frecuencia, quedarían las de altas frecuencias:

$$f(m,n) - f_{LPF}(m,n) = f_{HPF}(m,n) \qquad (3-24)$$

La fórmula queda entonces:

$$f_{HB}(m,n) = (A-1) * f(m,n) + f_{HPF}(m,n) \qquad (3-25)$$

$$f_{HB}(m,n) = (A-1) * f(m,n) + f(m,n) * h_{HPF}(m,n) \qquad (3-26)$$

donde $h_{HPF}(m,n)$ es el filtro paso alto.

La implementación del filtro de refuerzo de frecuencias altas en una imagen se logra aplicando una máscara adecuada y realizando la operación de convolución.

Algunas de las máscaras típicas usada para esta técnica son [13]:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & A+4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ -1 & A+8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad (3-27)$$

Estas dos máscaras las vamos a usar para ver cómo se comportan, tomamos el factor de amplificación como $A = 1$, llamaremos a la primera mask1 y a la segunda mask2.

3.3.4 Comparativa de los métodos

Tabla 3. Comparativa de técnicas de mejora de la imagen.

	Ventajas	**Desventajas**
Ecualización del histograma	Consigue mejorar el contrate de la imagen con una distribución uniforme del histograma. Es un método sencillo y automático. Útil en imágenes médicas, como radiografías y TAC, ya que permite separar regiones óseas, musculares, tejidos y órganos.	No funciona bien con imágenes con histogramas muy irregulares, ya que no se consigue hacerlo del todo uniforme. Puede producir efectos no deseados cuando la profundidad de color es pequeña.
Unsharp masking	El enmascaramiento del desenfoque es una técnica bastante útil para aumentar la nitidez de una imagen a través de la aplicación de una máscara. Posee parámetros para ajustar el método y hacerlo así más flexible.	Puede crear efectos de borde llamativos no deseados. Puede aumentar el ruido de los píxeles de la imagen. Esta limitado por un factor, si se sobrepasa, se trataría entonces de un filtro High Boost.
High Boost Filter	Es una técnica para la mejora, sobre todo, de la nitidez de la imagen, al igual que el enmascaramiento del desenfoque. Produce un efecto más agresivo que el enmascaramiento del desenfoque debido a que el factor de amplificación es mayor (≥ 1). Funciona mejor en imágenes, regiones o áreas oscuras.	El principal inconveniente es que la amplificación afecta tanto a los píxeles de información como los píxeles de ruido. Se aumenta el ruido de la imagen.

4 IMPLEMENTACIÓN EN MATLAB

Se desarrolla ahora, en la aplicación MATLAB, los métodos usados para la eliminación del ruido y mejora de las imágenes.

4.1 Aplicación de los filtros de media y mediana en MATLAB

En este apartado vamos a indicar el código, variables, métodos, etc, usados en MATLAB para eliminar el ruido gaussiano y de sal y pimienta, a través de los filtros de media y mediana.

```
i1 = 'ISIC_0498990.jpg';
i2 = 'ISIC_0984817.jpg';
i4 = 'ISIC_4602463.jpg';
i5 = 'ISIC_6821316.jpg';

f=imread(i1);                    %Leemos la imagen que deseemos
f_gray=rgb2gray(f);              %Imagen original en escala de grises

%Ruido Gaussiano con media 0 y varianza 0.01
fng_gray=imnoise(f_gray,"gaussian",0,0.01);
fng=imnoise(f,"gaussian",0,0.01);

%Añadimos ruido sal & pimienta, d=densidad (por defecto es 0.05)
fni_gray=imnoise(f_gray,"salt & pepper");
fni=imnoise(f,"salt & pepper");

%Filtro de media sobre la imagen original con ruido gaussiano y ruido sal y pimienta.
%Usamos ahora imfilter para aplicar el filtro h(creado por fspecial, el cual será de
media)
h=fspecial('average');
fmedia_gray=imfilter(fng_gray,h);
fmedia=imfilter(fng,h);

%Ahora con ruido sal y pimienta
fmediaS_gray=imfilter(fni_gray,h);
fmediaS=imfilter(fni,h);

%% Aplicamos el filtro de mediana sobre la imagen con ruido gaussiano.
fmedianaGauss_gray = medfilt2(fng_gray);     %en escala de grises
%En cada plano R, G y B y encadenamos al final
fmedianaGauss_R = medfilt2(fng(:,:,1));
fmedianaGauss_G = medfilt2(fng(:,:,2));
fmedianaGauss_B = medfilt2(fng(:,:,3));
fmedianaGauss_color = cat(3,fmedianaGauss_R,fmedianaGauss_G,fmedianaGauss_B);

%% Aplicamos el filtro de mediana sobre la imagen con ruido impulsivo.
fmedianaSal_gray = medfilt2(fni_gray);
%En cada plano R, G y B y encadenamos al final
fmedianaSal_R = medfilt2(fni(:,:,1));
fmedianaSal_G = medfilt2(fni(:,:,2));
fmedianaSal_B = medfilt2(fni(:,:,3));
```

```
fmedianaSal_color = cat(3,fmedianaSal_R,fmedianaSal_G,fmedianaSal_B);



%% Comparativas en imágenes
figure('Name','COMPARATIVA COLOR RUIDO GAUSSIANO')
subplot(1,4,1),imshow(f);                    %title('Imagen original')
subplot(1,4,2),imshow(fng);                  %title('Imagen con ruido gaussiano')
subplot(1,4,3),imshow(fmediaG);              %title('Filtro de media')
subplot(1,4,4),imshow(fmedianaGauss_color);  %title('Filtro de mediana')

figure('Name','COMPARATIVA COLOR RUIDO SAL Y PIMIENTA')
subplot(1,4,1),imshow(f);               %title('Imagen original')
subplot(1,4,2),imshow(fni);             %title('Imagen con ruido sal y pimienta')
subplot(1,4,3),imshow(fmediaS);         %title('Filtro de media')
subplot(1,4,4),imshow(fmedianaSal_color);%title('Filtro de mediana')

figure('Name','COMPARATIVA RUIDO GAUSSIANO')
subplot(1,4,1),imshow(f_gray);               %title('Imagen original')
subplot(1,4,2),imshow(fng_gray);             %title('Imagen con ruido gaussiano')
subplot(1,4,3),imshow(fmediaG_gray);         %title('Filtro de media')
subplot(1,4,4),imshow(fmedianaGauss_gray);   %title('Filtro de mediana')

figure('Name','COMPARATIVA RUIDO SAL&PIMIENTA')
subplot(1,4,1),imshow(f_gray);           %title('Imagen original')
subplot(1,4,2),imshow(fni_gray);         %title('Imagen con ruido sal y pimienta')
subplot(1,4,3),imshow(fmediaS_gray);     %title('Filtro de media')
subplot(1,4,4),imshow(fmedianaSal_gray); %title('Filtro de mediana')
```

- Funciones de MATLAB usadas:

imread(filename):
Esta función lee la imagen del archivo especificado por el atributo de entrada filename, además tiene otros atributos de entrada que permiten especificar el formato del archivo (fmt), o incluso leer una imagen o imágenes de un grupo (idx).

rgb2gray(RGB):
Este método convierte la imagen en color en el espacio RGB introducida por parámetro de entrada a una imagen en escala de grises. Por tanto, elimina la información de tono y saturación y conserva la de luminosidad.

imnoise(I, 'tipo de ruido'):
Añade el tipo de ruido indicado como parámetro de entrada a la imagen I, normalizando los valores entre 0 y 1. Además según los tipos de ruido, se pueden especificar otros parámetros, señalamos algunos:

- imnoise(I, 'gausssian', m, var_gauss): Introduce ruido gaussiano con media m y varianza var_gauss, por defecto la media vale 0 y la varianza 0.01.
- imnoise(I, 'salt & pepper', d): Añade ruido de sal y pimienta, donde d equivale a la densidad de ruido, por defecto su valor es de 0.05, que quiere decir que afecta solo a 5% de los píxeles.
- imnoise(I, 'poisson'): Genera ruido de disparo a partir de los datos en lugar de añadir ruido artificial a los datos.

fspecial (type):

Esta función crea un filtro en 2 dimensiones del tipo especificado como parámetro de entrada, el valor devuelto por fspecial actúa como núcleo de correlación, que es la forma adecuada para utilizar con otro método que describiremos a continuación, imfilter. Según el tipo de filtro, además admite unos atributos u otros:

- fspecial ('average', hsize): Genera un filtro promediador de tamaño hsize, si no se especifica el tamaño, por defecto es 3 (3x3).
- fspecial ('disk', radius): devuelve un filtro promediador circular situado dentro de la matriz de tamaño 2*radius+1.

imfilter (I, h):
Función que realiza el filtrado multidimensional de la imagen I a partir de un filtro indicado como parámetro de entrada, h.

medfilt2 (I):
Este método realiza el filtrado de mediana de la imagen I en 2 dimensiones. Por cada píxel de la imagen I, se obtiene un píxel resultante que contiene el valor de la mediana de un entorno 3x3 del píxel de entrada.

cat (d_N, A_1, A_2, … , A_N):
Función que permite concatenar elementos A_1, A_2, … , A_N en la dimensión d_N. La concatenación se realiza desde el último elemento hacia el primero.

Funciones para la representación gráfica de los resultados:

figure (Name, Value):
Crea una ventana para representar una figura (puede ser una imagen, una gráfica, resultados, etc). Puede no tener parámetros de entrada, pero se puede modificar los valores de la figura, indicándolo en el parámetro Name y el valor de éste.
Ejemplo: figure('Color', 'red'). Estos parámetros establecerán el color del fondo como rojo.

subplot (n, m, p):
Sirve para crear ejes en posiciones segmentadas, divide la figura en una cuadrícula de m por n y crea ejes en la posición especificada por p.

imshow (I):
Permite mostrar la imagen I. Utiliza el intervalo de visualización predeterminado para el tipo de datos de la imagen y optimiza la figura, los ejes y las propiedades del objeto de imagen para su visualización. Si se incluye [] como parámetro, representa la imagen entre los valores máximos y mínimos.

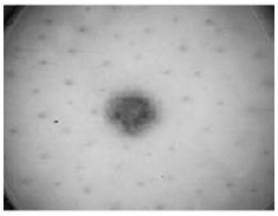
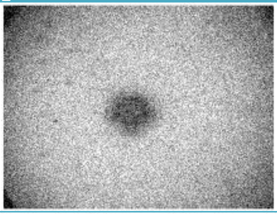
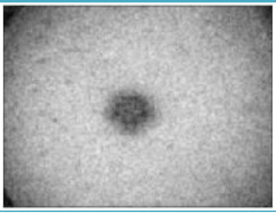
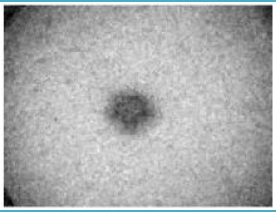
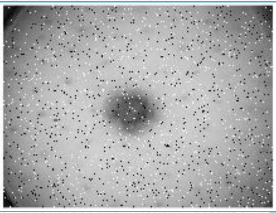
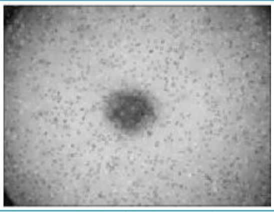
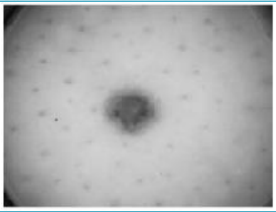
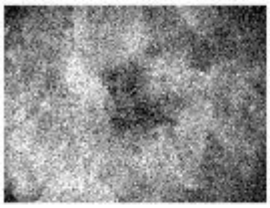
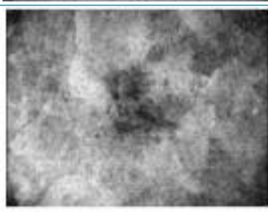
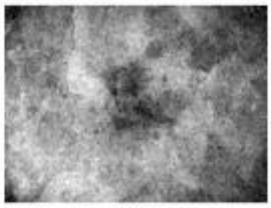
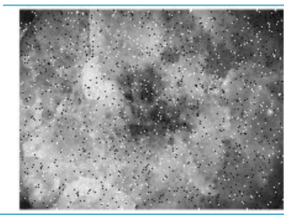
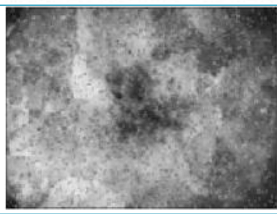
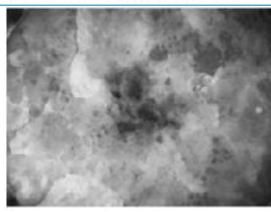
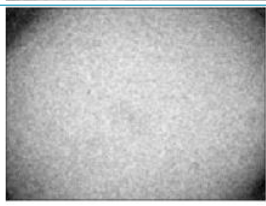
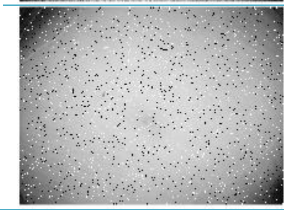
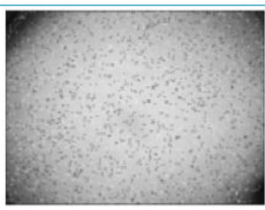
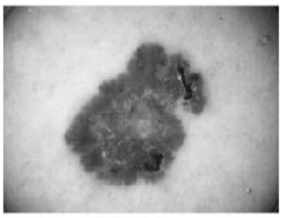
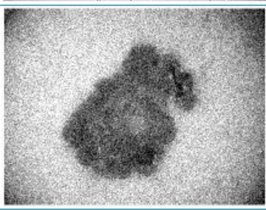
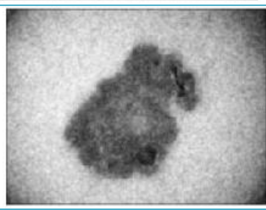
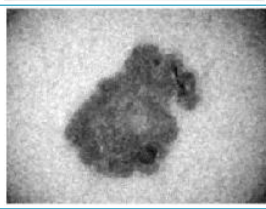
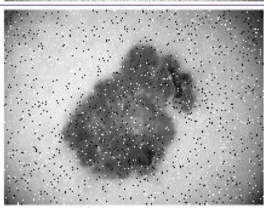
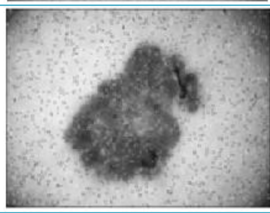
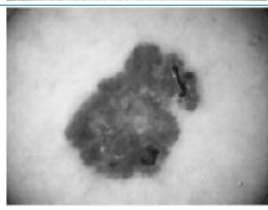
4.1.1 Comparativa en imágenes.

Vamos a ver ahora, representado en la Tabla 4 y Tabla 5, el resultado del apartado anterior.

Se observa cómo se comportan tanto el filtro de media cómo el de mediana en imágenes con ruidos gaussianos y ruidos impulsivos, tanto en escala de grises como en color. Usamos algunas de las imágenes del estudio, descargadas de la base de datos de imágenes médicas ISIC.

- Escala de grises (Plano intensidad):

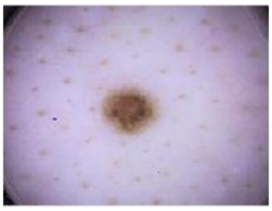
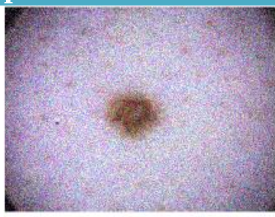
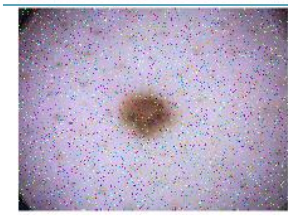
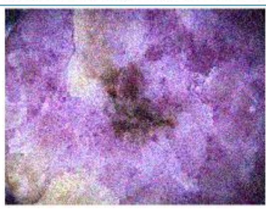
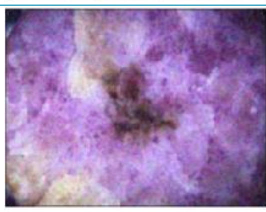
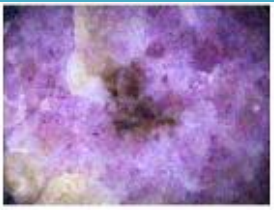
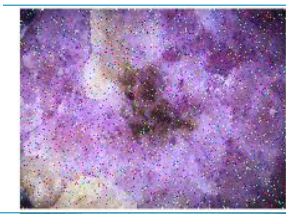
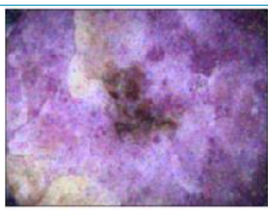
Tabla 4. Comparativa en imágenes, escala de grises.

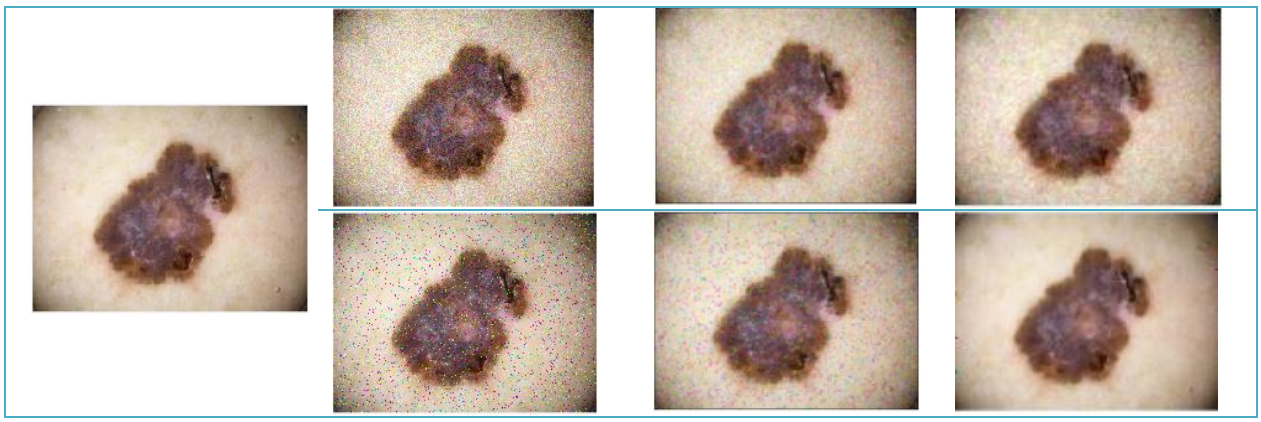
Imagen original	Imagen con ruido Gaussiano/Sal y pimienta	Filtro de media	Filtro de mediana
			
			
			
			
			
			
			
			

A simple vista, observamos que para para la imagen distorsionada con ruido gaussiano funciona bastante mejor el filtro de media que el filtro de mediana. Y para la imagen distorsionada con ruido sal y pimienta, funciona mucho mejor el filtro de la mediana que el de media.
En el apartado 4.6 de este capítulo, esta evidencia además se verá plasmada con las medidas de calidad de la imagen, MSE y PSNR.

- En color:

Tabla 5. Comparativa en imágenes a color.

Imagen original	**Imagen con ruido Gaussiano/Sal y pimienta**	**Filtro de media**	**Filtro de mediana**
			
			
			
			
			
			



En ambas tablas, y aunque viene indicado en el título de la columna, la primera fila de cada imagen corresponde al tratamiento del ruido gaussiano y la siguiente fila al tratamiento del ruido de sal y pimienta, así sucesivamente para cada imagen.

De la misma forma, en color, vemos que el ruido sal y pimienta se elimina mejor con el filtro de mediana que con el filtro de media. Y para el ruido gaussiano, es el filtro de media el más efectivo.

Analizaremos en el apartado 4.6 estas conclusiones con las medidas de fidelidad de la imagen, MSE y PSNR.

4.2 Binarización del histograma en MATLAB

Vamos a mostrarlo en un ejemplo realizado en MATLAB para la imagen: ISIC_0498990.jpg, obtenida de la base de datos de ISIC.

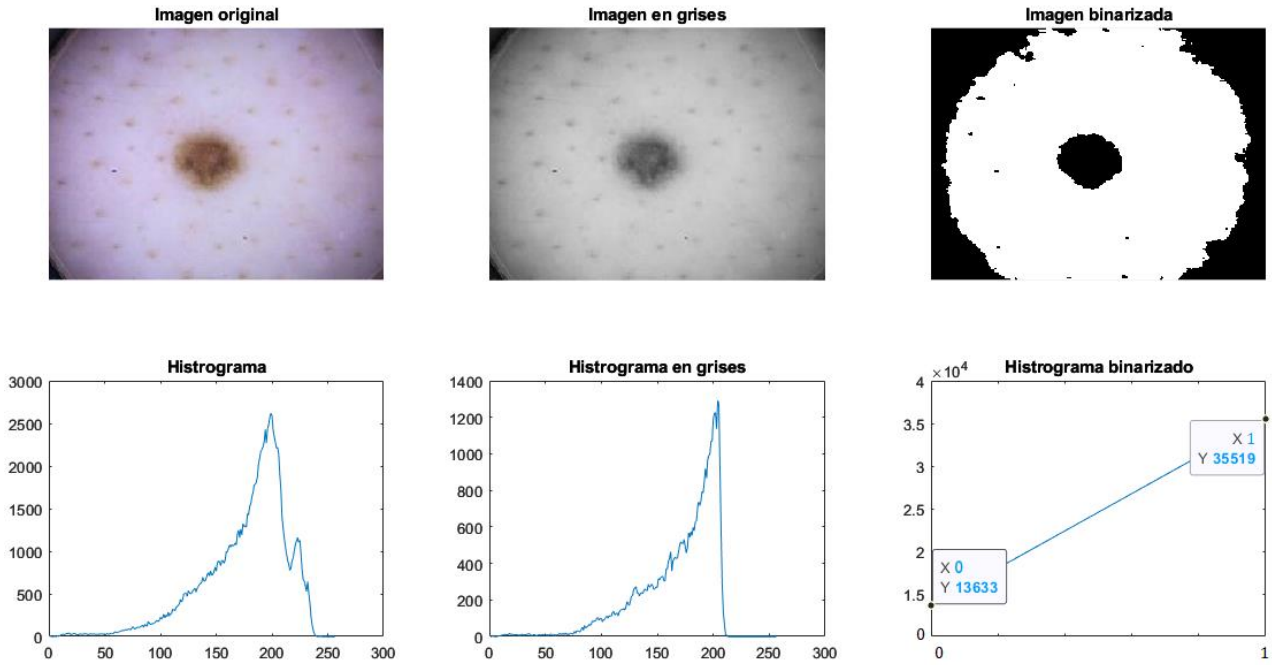


Figura 14. Binarización del histograma. Imágenes con sus correspondientes histogramas debajo.

Donde se aprecia las regiones binarizadas a blanco y negro claramente en la imagen, y el número de valores de blanco y negro en el histograma.

Realizado con el siguiente código:

```
i1 = 'ISIC_0498990.jpg';

f=imread(i1);          %Leemos las imágenes
fgray=rgb2gray(f);     %Convertimos de RBG a escala de grises
fB=imbinarize(fgray);  %Binarizamos la imagen en escala de grises

%Obtenemos los histogramas de la imagen
h= imhist(f);
hg= imhist(fgray);
hb=imhist(fB);

%Funciones que obtienen el valor del umbral, normalizado entre [0 1]
To = otsuthresh(hg)
Tg = graythresh(fgray)

%Representación de los resultados
figure('Name','Imagenes Binarizada')
subplot(2,3,1),imshow(f);title('Imagen original')
subplot(2,3,2),imshow(fgray);title('Imagen en grises')
subplot(2,3,3),imshow(fB);title('Imagen binarizada')
subplot(2,3,4),plot(h);title('Histrograma')
subplot(2,3,5),plot(hg);title('Histrograma en grises')
subplot(2,3,6),plot(hb);title('Histrograma binarizado')
```

Descripciones de las funciones usadas aún no mencionadas:

imhist (I):
Calcula el histograma de la imagen I . Devuelve los recuentos del histograma.

imbinarize (I):
Crea una imagen binaria a partir de otra en 2 ó 3 dimensiones, seleccionando un valor umbral, de manera que todos los valores por debajo de ese umbral, le asignará un 0 (negro); y los que estén por encima de ese umbral le asigna un 1 (blanco). Por defecto utiliza el método de Otsu para obtener ese umbral, pero se puede indicar por parámetros un valor concreto.

otsuthresh (counts):
Calcula un umbral global a partir de los recuentos de un histograma, counts, usando el método de Otsu, el cual elige el que minimiza la varianza interclase de los píxeles blancos y negros pasados por el umbral.

graythresh (I):
Calcula un umbral global a partir de una imagen en escala de grises I empleando el método de Otsu.

plot (G):
Representa en una gráfica los nodos y los bordes.

4.3 Ecualización del histograma en MATLAB

A continuación, vamos a representar en la Tabla 6, Tabla 7 y Tabla 8, ejemplos de imágenes en escala de grises y en color, de la base de datos ISIC, junto con sus histogramas, originales y ecualizados.

En el caso de las imágenes en color, voy a aplicar el método de ecualización del histograma de dos maneras diferentes. La primera se va a aplicar a cada plano r, g y b; la segunda, transformamos a un espacio de color donde exista el plano de intensidad, como por ejemplo V de HSV (*Hue, Saturation, Value).*

Tabla 6. Ecualización del histograma en escala de grises.

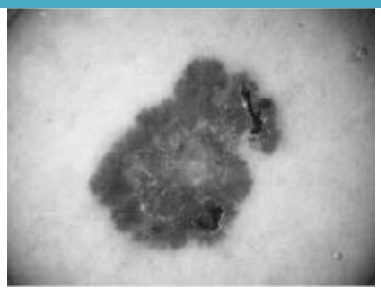
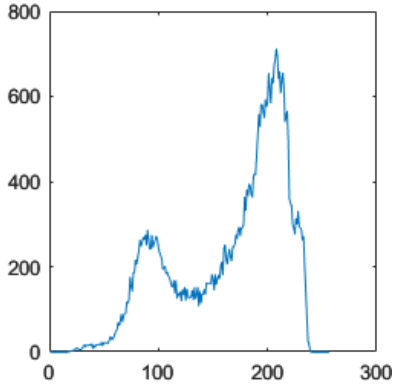
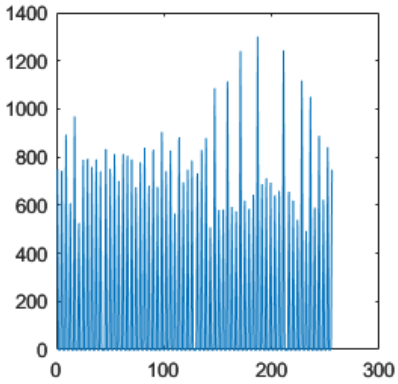
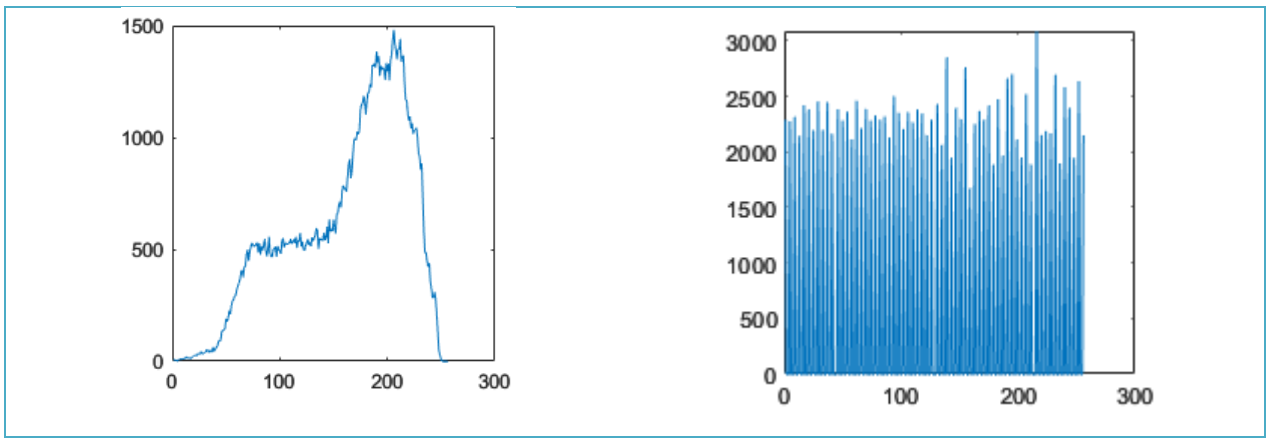
Imagen en escala de grises	Imagen en escala de grises ecualizada
	
Histograma en escala de grises	**Histograma en escala de grises ecualizado**
	

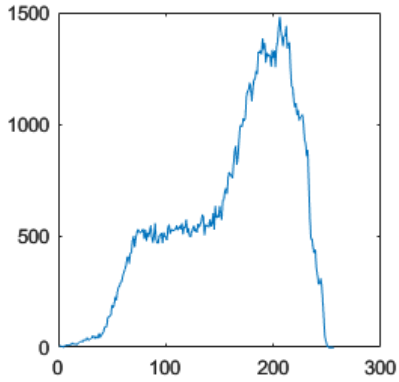
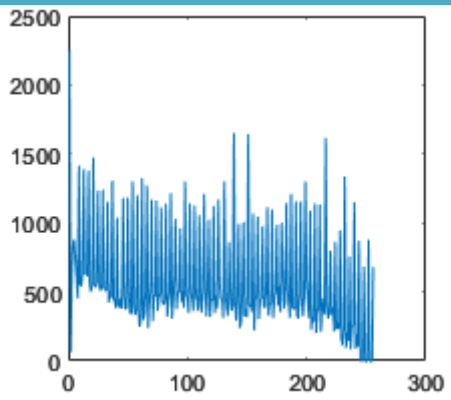
Tabla 7. Ecualización del histograma en cada plano R, G y B.

Imagen original	Imagen ecualizada en cada plano R, G y B
	
Histograma original	**Histograma ecualizado**



En la Tabla 7 se observa que, al aplicar la ecualización del histograma a cada plano de color, su función densidad de probabilidad es más o menos uniforme, pero se introducen artefactos de color en la imagen por lo que se distorsiona. Por ello la ecualización del histograma se aplica a los planos de intensidad únicamente.

Tabla 8. Ecualización del histograma al plano V.

Imagen original	Imagen ecualizada en el plano V
	
Histograma original	**Histograma ecualizado**
	

Al realizar el método en el plano V, tal y como aparece en la Tabla 8, visualizamos la imagen con mayor contraste. Esta es la manera correcta de realizar la ecualización en una imagen en color.

Programación en Matlab:

```
%Ecualización del histograma
i5 = 'ISIC_6821316.jpg';
f=imread(i5);              %Leemos las imágenes
fgray= rgb2gray(f);        %Convertimos la imagen en escala de grises

eg=histeq(fgray);       %Ecualizamos la imagen en escala de grises
```

```
%Obtenemos los histogramas de las imágenes originales y la de escala de grises
h= imhist(f);
hg= imhist(fgray);
hge=imhist(eg);

%% Ecualizar cada componente R, G y B
%Separamos la imagen en los planos RGB
r=f(:,:,1); g=f(:,:,2); b=f(:,:,3);

%Ecualizamos cada componente
g1=histeq(r);
g2=histeq(g);
g3=histeq(b);
gfin=cat(3,g1,g2,g3);       %Encadenamos las componentes ecualizadas
hRGB= imhist(gfin);         %Histograma ecualizado
% Ecualización del histograma pasando al espacio HSV
fHSV=rgb2hsv(f);            %Convertimos al espacio HSV
feHSV=histeq(fHSV(:,:,3));  %Ecualizamos solo la componente V
feHSVfin=cat(3,fHSV(:,:,1),fHSV(:,:,2),feHSV); %Encadenamos las componentes
fHSV_RGB=hsv2rgb(feHSVfin); %Convertimos de nuevo a RGB
hHSV_RGB= imhist(fHSV_RGB); %Histograma ecualizado

%Representación gráfica
figure('Name','Ecualizacion del histograma')
subplot(2,5,1),imshow(f);title('Imagen original')
subplot(2,5,2),imshow(fgray);title('Imagen en grises')
subplot(2,5,3),imshow(eg);title('Imagen en grises ecualizada')
subplot(2,5,4),imshow(gfin);title('Imagen ecualizada en cada plano RGB')
subplot(2,5,5),imshow(fHSV_RGB);title('Imagen RGB ecualizada en plano V')

subplot(2,5,6),plot(h);title('Histograma original')
subplot(2,5,7),plot(hg);title('Histograma original en grises')
subplot(2,5,8),plot(hge);title('Histograma en grises ecualizado')
subplot(2,5,9),plot(hRGB);title('Histograma RGB ecualizado')
subplot(2,5,10),plot(hHSV_RGB);title('Histograma V ecualizado')
```

histeq (I):
Función que mejora el contraste mediante la ecualización del histograma de la imagen introducida por parámetro I, solo en el plano de intensidad. Se pueden además especificar en la entrada, el número de niveles de intensidad.

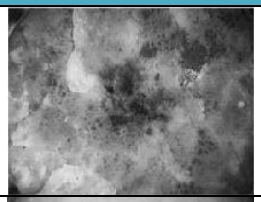
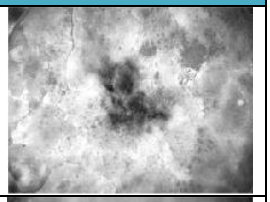
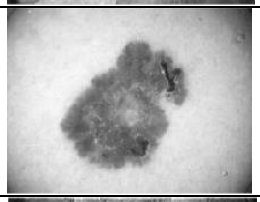
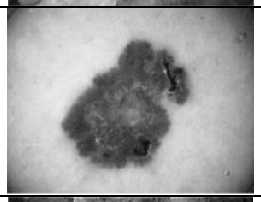
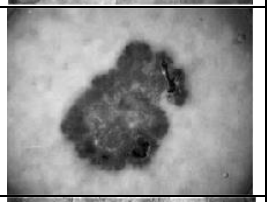
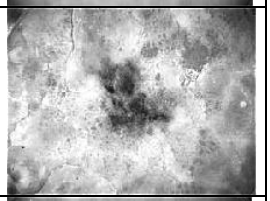
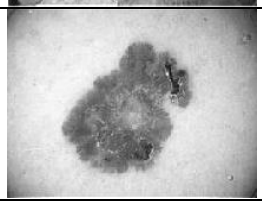
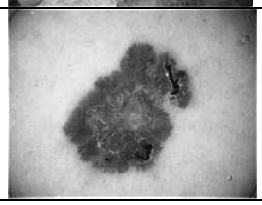
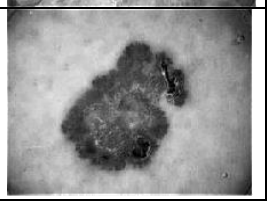
rgb2hsv (I):
Convierte los plano R, G y B de una imagen I a los planos H(*hue*), S(*saturation*) y V(*value*).

hsv2rgb (I):
Este método realiza la operación inversa al anterior, es decir, convierte una imagen en el espacio HSV al espacio RGB.

4.4 Enmascaramiento del desenfoque en MATLAB

Vamos a ver, con imágenes de la base de datos ISIC, cómo funciona el método de enmascaramiento del desenfoque usando el software MATLAB.
En este caso, aplicamos el método, con la máscara 3x3 indicada en (3-21) a cada plano de color RGB por separado.

Tabla 9. Unsharp masking, RGB.

	Conjunta	Plano R	Plano G	Plano B
Imágenes originales				
				
Imágenes tras aplicar Unsharp masking				
				

A continuación, muestro el código programado:

```
i2 = 'ISIC_0984817.jpg';
i5 = 'ISIC_6821316.jpg';

f=imread(i2);          %Leemos la imagen
fdouble= im2double(f); %Transformamos la imagen a double y normalizada

%Separamos los planos RGB
r=fdouble(:,:,1);
g=fdouble(:,:,2);
b=fdouble(:,:,3);

%Definimos la máscara y aplicamos su filtrado con filter2() a cada componente RGB
mask=[-1/8 -1/8 -1/8;-1/8 2 -1/8;-1/8 -1/8 -1/8];
g1fil=filter2(mask,r);
g2fil=filter2(mask,g);
g3fil=filter2(mask,b);
gfin=cat(3,g1fil,g2fil,g3fil); %Encadenamos las 3 componentes a la imagen

%Representación de los resultados
figure('Name','Unsharp masking')
subplot(2,4,1),imshow(fdouble);title('Imagen original')
subplot(2,4,2),imshow(r);title('Plano R')
subplot(2,4,3),imshow(g);title('Plano G')
subplot(2,4,4),imshow(b);title('Plano B')
subplot(2,4,5),imshow(gfin);title('Imagen USM')
subplot(2,4,6),imshow(g1fil);title('Plano R')
subplot(2,4,7),imshow(g2fil);title('Plano G')
subplot(2,4,8),imshow(g3fil);title('Plano B')
```

Funciones usadas:

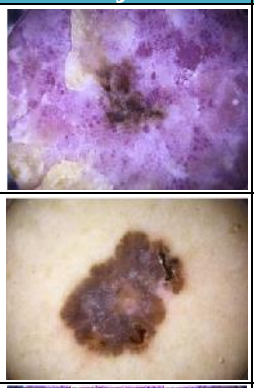
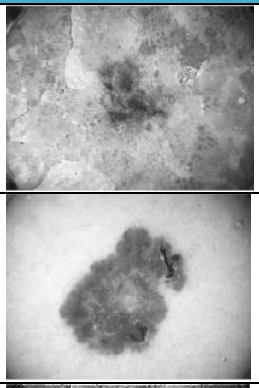
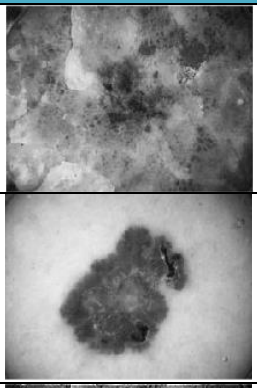
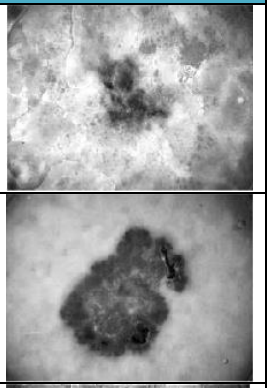
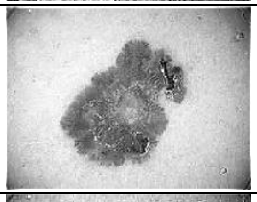
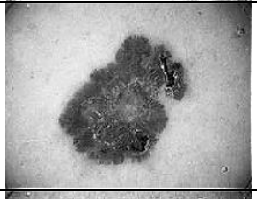
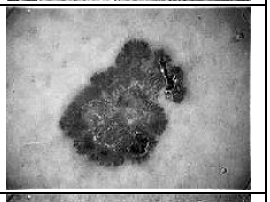
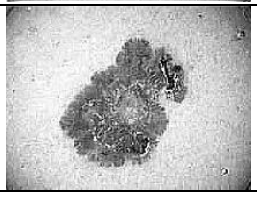
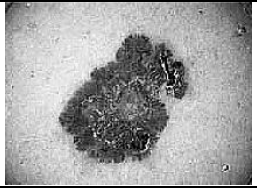
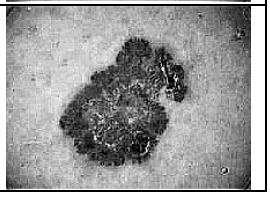
im2double (I):
Convierte la imagen I en formato de doble precisión (*double*). Además, escala la salida de los tipos de datos enteros a valores entre [0,1].

filter2 (H, X):
Crea un filtro en 2 dimensiones. Aplica un filtro de respuesta de impulso finito a una matriz de datos X según los coeficientes de la matriz H.

Es necesario convertir la imagen a formato *double* ya que al filtrar con la máscara 3x3 en el filtro, toma ese formato.

4.5 Filtro de refuerzo de frecuencias altas en MATLAB

Tabla 10. High Boost filtering, RGB.

	Conjunta	Plano R	Plano G	Plano B
Imágenes originales				
Imágenes tras aplicar High Boost filtering				
				
				
				

En la tabla 10, la primera fila de “Imágenes tras aplicar *High Boost filtering*” corresponde al aplicar la máscara *mask1* y la segunda fila a *mask2*, y así sucesivamente para las demás.

Se puede observar cómo, sobre todo, los bordes de los elementos de las imágenes se ven más nítidos, incluso más que con la técnica de enmascaramiento del desenfoque, ya que estamos aplicando aquí un factor de amplificación mayor.
Funciona mejor en zonas oscuras, pero también sucede que, si existe ruido en los píxeles, al amplificarlo, también se amplifica el ruido.

Muestro el código programado:

```
i2 = 'ISIC_0984817.jpg';
i5 = 'ISIC_6821316.jpg';

f=imread(i5);          %Leemos las imágenes
fdouble= im2double(f); %Transformamos la imagen a doble, normaliza entre [0,1]

%Separamos los planos RGB
r=fdouble(:,:,1);
g=fdouble(:,:,2);
b=fdouble(:,:,3);

%Definimos la máscara de altas frecuencias
mask1=[0 -1 0;-1 5 -1;0 -1 0];
mask2=[-1 -1 -1;-1 9 -1;-1 -1 -1];

%Aplicamos su filtrado con filter2() a cada componente RGB
g1fil=filter2(mask1,r);
g2fil=filter2(mask1,g);
g3fil=filter2(mask1,b);
gfin=cat(3,g1fil,g2fil,g3fil); %Encadenamos las 3 componentes a la imagen

%Representación de los resultados
figure('Name','High Boost filtering')
subplot(2,4,1),imshow(fdouble);title('Imagen original')
subplot(2,4,2),imshow(r);title('Plano R')
subplot(2,4,3),imshow(g);title('Plano G')
subplot(2,4,4),imshow(b);title('Plano B')
subplot(2,4,5),imshow(gfin);title('Imagen High Boost')
subplot(2,4,6),imshow(g1fil);title('Plano R')
subplot(2,4,7),imshow(g2fil);title('Plano G')
subplot(2,4,8),imshow(g3fil);title('Plano B')
```

Como en la técnica anterior, se ha aplicado el filtrado de manera individual a cada uno de los planos de color R, G, y B. Posteriormente, se ha concatenado los resultados para ver la imagen compuesta tras el filtrado.

4.6 Implementación de medidas referenciadas

Definimos en el capítulo 2 qué eran las medidas referenciadas, para la cual es necesario la comparación con la imagen original, por tanto, en este punto, vamos a implementar dichas medidas con el software MATLAB, comparando imágenes originales con imágenes con ruido y con imágenes que han filtrado dichos ruidos.

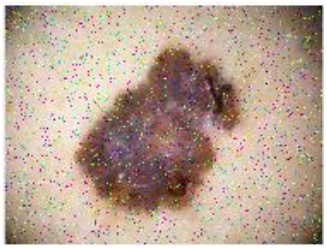
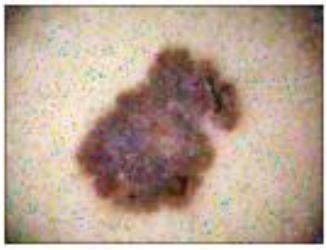
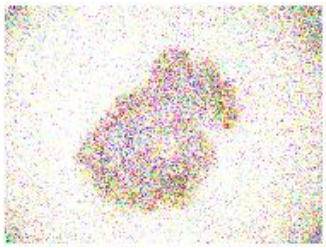
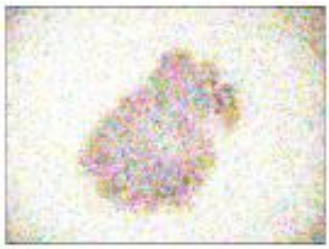
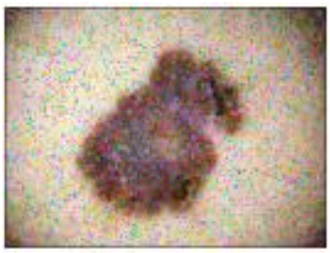
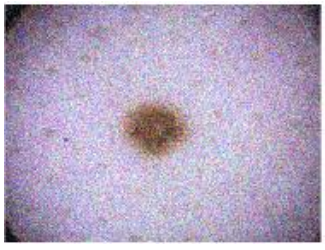
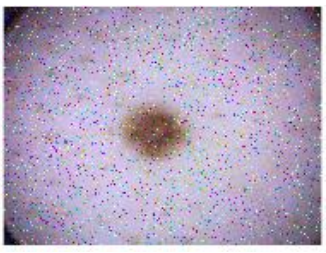
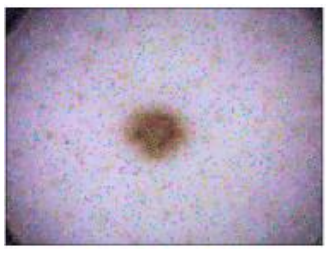
4.6.1 Con filtro de media

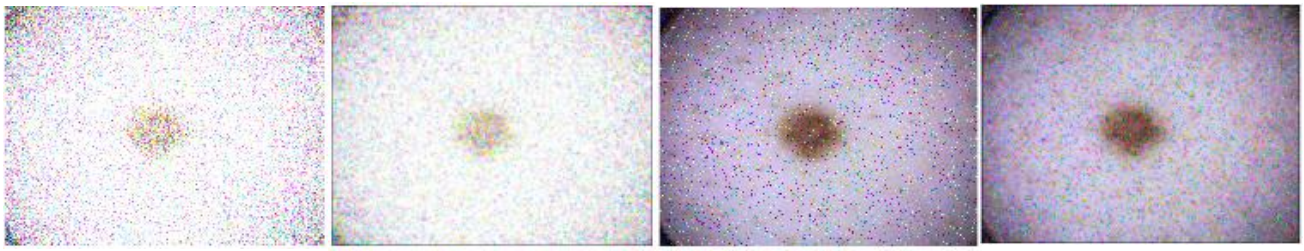
Mostraremos en este apartado dos tablas, Tabla 11 y Tabla 12, las cuales representan los resultados de las medidas referenciadas MSE y PSNR, la primera solo aplicando las transformaciones necesarias al plano L del espacio CIE, convirtiendo después al espacio RGB; la segunda aplicando la máscara a cada plano de color RGB.

Vamos a comparar la imagen original con la misma imagen pero añadiéndole ruido gaussiano o de sal y pimienta. Y la imagen original con la imagen tras el filtrado de media, a cada caso.

Los valores de la MSE están normalizado entre 0 y 1.
Los valores de la PSNR están calculados según el rango dinámico de las imágenes, en este caso está entre 0 y 255.

Tabla 11. MSE y PSNR. Filtro de media, plano L.

Imagen con ruido gaussiano	Imagen con filtro de media al ruido gaussiano	Imagen con ruido de sal y pimienta	Imagen con filtro de media al ruido sal y pimienta
			
media = 0 ; varianza = 0,01 mse = 0,0285 psnr = 63,5842	mse = 0,0272 psnr = 63,7896	densidad = 0,05 mse = 0,0471 psnr = 61,3968	mse = 0,0337 psnr = 62,8591
			
media = 0,5 ; varianza = 0,1 mse = 0,3300 psnr = 52,9455	mse = 0,2829 psnr = 53,6139	densidad = 0,1 mse = 0,0929 psnr = 58,4497	mse = 0,0635 psnr = 60,1061
			
media = 0 ; varianza = 0,01 mse = 0,0290 psnr = 63,5117	mse = 0,0264 psnr = 63,9174	densidad = 0,05 mse = 0,0453 psnr = 61,5693	mse = 0,0323 psnr = 63,0403



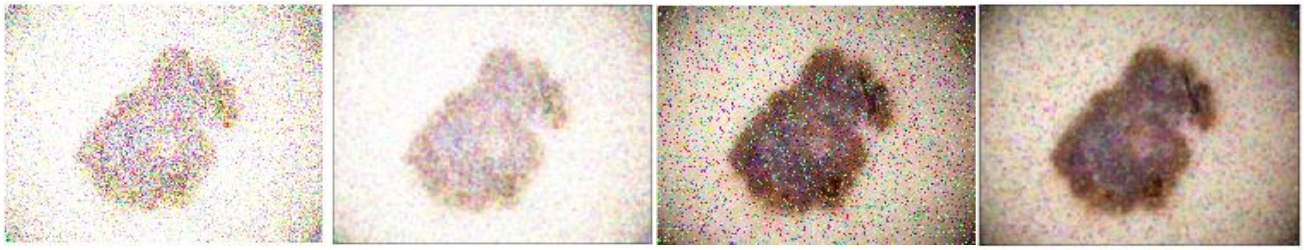
media = 0,5 ; varianza = 0,1
mse = 0,2555
psnr = 54,0577

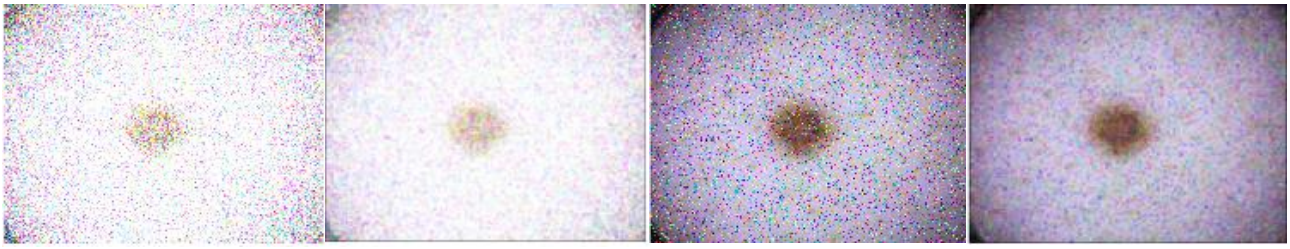
mse = 0,2094
psnr = 54,9207

densidad = 0,1
mse = 0,0913
psnr = 58,5260

mse = 0,0628
psnr = 60,1501

Tabla 12. MSE y PSNR. Filtro de media en RGB.

Imagen con ruido gaussiano	Imagen con filtro de media al ruido gaussiano	Imagen con ruido de sal y pimienta	Imagen con filtro de media al ruido sal y pimienta
			
media = 0 ; varianza = 0,01 mse = 0,0285 psnr = 63,5842	mse = 0,0055 psnr = 70,7576	densidad = 0,05 mse = 0,0471 psnr = 61,3968	mse = 0,0080 psnr = 69,0740
			
media = 0,5 ; varianza = 0,1 mse = 0,3300 psnr = 52,9455	mse = 0,2829 psnr = 53,6139	densidad = 0,1 mse = 0,0929 psnr = 58,4497	mse = 0,0143 psnr = 66,5829
			
media = 0 ; varianza = 0,01 mse = 0,0290 psnr = 63,5117	mse = 0,0049 psnr = 71,2588	densidad = 0,05 mse = 0,0453 psnr = 61,5693	mse = 0,0071 psnr = 69,6386



media = 0,5 ; varianza = 0,1
mse = 0,2555
psnr = 54,0577

mse = 0,2040
psnr = 55,0352

densidad = 0,1
mse = 0,0913
psnr = 58,5260

mse = 0,0133
psnr = 66,8998

Desarrollado con este código:

```
i1 = 'ISIC_0498990.jpg';
i5 = 'ISIC_6821316.jpg';
f=imread(i1);
f=im2double(f); %Normalizamos los valores entre 0 y 1
%Añadimos ruido blanco gaussiano de media y varianza indicada por parámetros
fng=imnoise(f,"gaussian",0.5,0.1);
%Añadimos ruido impulsivo o salt & pepper, d=densidad %
fni=imnoise(f,"salt & pepper",0.1);
%% División de la imagen con ruido gaussiano y ruido sal y pimienta en los canales
RGB%
fngR=fng(:,:,1); fngG=fng(:,:,2); fngB=fng(:,:,3);
fniR=fni(:,:,1); fniG=fni(:,:,2);  fniB=fni(:,:,3);

%% Aplicamos el filtro (con la función fspecial) sobre la imagen original con ruido
gaussiano y ruido sal y pimienta.
% Usamos ahora imfilter para aplicar el filtro h(creado por fspecial, el cual será de
media)
h=fspecial('average');
fmediaG=imfilter(fng,h);
fmediaS=imfilter(fni,h);
%imfilter a cada plano:
%Para el ruido gaussiano
fmediaGaussR=imfilter(fng(:,:,1),h);
fmediaGaussG=imfilter(fng(:,:,2),h);
fmediaGaussB=imfilter(fng(:,:,3),h);
fmediaGaussf=cat(3,fmediaGaussR,fmediaGaussG,fmediaGaussB); %encadenamos los planos
%Para el ruido de sal y pimienta
fmediaSalR=imfilter(fni(:,:,1),h);
fmediaSalG=imfilter(fni(:,:,2),h);
fmediaSalB=imfilter(fni(:,:,3),h);
fmediaSalf=cat(3,fmediaSalR,fmediaSalG,fmediaSalB); %encadenamos los planos

%Pintamos los resultados%
figure('Name','Filtro de media con fspecial a img original')
subplot(3,3,[1,4]),imshow(f); title('Imagen original')
subplot(3,3,2),imshow(fng);title('Imagen con ruido gaussiano')
subplot(3,3,3),imshow(fni);title('Imagen con ruido sal y pimienta')
subplot(3,3,5),imshow(fmediaGaussf);title('Imagen filtrada por plano de ruido
gaussiano')
subplot(3,3,6),imshow(fmediaSalf);title('Imagen filtrada por plano de ruido sal y
pimienta')
```

```
%% Aplicamos el filtro de media al plano L.
% Convertimos la imagen de RGB a CIE. Imagen Original.
fCIE=rgb2lab(f);   %Al cambiar del espacio rgb a lab, cambia el rango dinámico de la
imagen entre 0 y 100, por eso después lo representamos entre esos valores.
fCIEL=fCIE(:,:,1); % Obtenemos el plano L (escala de grises).

% Imagen con ruido GAUSSIANO
fngCIE=rgb2lab(fng);
fngCIEL=fngCIE(:,:,1);   % Obtenemos el plano L (escala de grises).
% Imagen con ruido SAL Y PIMIENTA
fniCIE=rgb2lab(fni);
fniCIEL=fniCIE(:,:,1);   % Obtenemos el plano L (escala de grises).

% Imagen filtrada de ruido gaussiano (filtro de media) al plano L
fmediaCIEG=imfilter(fngCIEL,h);
fmediaCIEGf=cat(3,fmediaCIEG,fngCIE(:,:,2),fngCIE(:,:,3)); %Encadenamos los planos.
fmediaCIEG_RGB=lab2rgb(fmediaCIEGf); %Volvemos al espacio RGB
fmediaCIEG_RGB(fmediaCIEG_RGB > 1.0 ) = 1;%Umbralizamos los valores max y min a 0 y 1
fmediaCIEG_RGB(fmediaCIEG_RGB < 0 ) = 0;

% Imagen filtrada de ruido sal y pimienta (filtro de media) al plano L
fmediaCIES=imfilter(fniCIEL,h);
fmediaCIESf=cat(3,fmediaCIES,fniCIE(:,:,2),fniCIE(:,:,3)); %Encadenamos los planos
fmediaCIES_RGB=lab2rgb(fmediaCIESf); %Al transformar otra vez a rgb se pierde el
rango dinamico, ¿qué hacer?
fmediaCIES_RGB(fmediaCIES_RGB > 1.0 ) = 1;
fmediaCIES_RGB(fmediaCIES_RGB < 0 ) = 0;

%% Representación en RGB tras media en el plano L de Lab. imshow(f,[]) cambia el
rango dinámico entre los valores máximo y mínimo de la imagen.
figure('Name','RGB To Lab 3 planos')
subplot(3,3,[1,4]),imshow(f,[]); title('Imagen original') %Obtenemos el plano L
subplot(3,3,2),imshow(fng,[]);title('Imagen con ruido gaussiano')
subplot(3,3,3),imshow(fni,[]);title('Imagen con ruido sal y pimienta')
subplot(3,3,5),imshow(fmediaCIEG_RGB,[]);title('Lab:Filtrado Gauss paso a RGB')
subplot(3,3,6),imshow(fmediaCIES_RGB,[]);title('Lab:Filtrado S&P paso a RGB')

%% ---------------------------- MEDIDADES DE FIDELIDAD -----------------------------
% Vamos calcular la MSE y la PSNR como medidas de fidelidad de las imágenes

%Calculamos el error cuadrático en el espacio RGB tras solo aplicar la transformación
al plano L
squaredErrorImageG = double((f(:,:,1)-fng(:,:,1)) .^ 2+(f(:,:,2)-fng(:,:,2)) .^
2+(f(:,:,3)-fng(:,:,3)) .^ 2);  %Entre la imagen original y con ruido gaussiano.
squaredErrorImageS = double((f(:,:,1)-fni(:,:,1)) .^ 2+(f(:,:,2)-fni(:,:,2)) .^
2+(f(:,:,3)-fni(:,:,3)) .^ 2);  %Entre la imagen original y con ruido sal y pimienta.

squaredErrorImageGf = double((f(:,:,1)-fmediaCIEG_RGB(:,:,1)) .^ 2+(f(:,:,2)-
fmediaCIEG_RGB(:,:,2)) .^ 2+(f(:,:,3)-fmediaCIEG_RGB(:,:,3)) .^ 2);    %Entre la
imagen original y filtrada del ruido gaussiano.
squaredErrorImageSf = double((f(:,:,1)-fmediaCIES_RGB(:,:,1)) .^ 2+(f(:,:,2)-
fmediaCIES_RGB(:,:,2)) .^ 2+(f(:,:,3)-fmediaCIES_RGB(:,:,3)) .^ 2);    %Entre la
imagen original y filtrada del ruido sal y pimienta.

%Error cuadrático en el espacio RGB, con ruido gaussiano y de sal y pimienta
squaredErrorImagecolorG = double((f(:,:,1)-fng(:,:,1)) .^ 2+(f(:,:,2)-fng(:,:,2)) .^
2+(f(:,:,3)-fng(:,:,3)) .^ 2);  %Entre la imagen original y con ruido gaussiano.
squaredErrorImagecolorS = double((f(:,:,1)-fni(:,:,1)) .^ 2+(f(:,:,2)-fni(:,:,2)) .^
2+(f(:,:,3)-fni(:,:,3)) .^ 2);  %Entre la imagen original y con ruido sal y pimienta.
```

```
squaredErrorImagecolorGf = double((f(:,:,1)-fmediaGaussf(:,:,1)) .^ 2+(f(:,:,2)-
fmediaGaussf(:,:,2)) .^ 2+(f(:,:,3)-fmediaGaussf(:,:,3)) .^ 2); %Entre la imagen
original y filtrada del ruido gaussiano
squaredErrorImagecolorSf = double((f(:,:,1)-fmediaSalf(:,:,1)) .^ 2+(f(:,:,2)-
fmediaSalf(:,:,2)) .^ 2+(f(:,:,3)-fmediaSalf(:,:,3)) .^ 2); %Entre la imagen original
y filtrada del ruido sal y pimienta

% Obtenemos los píxeles de la imagen.
[rows columns plane] = size(f); %Dividimos en filas, columnas y planos.

%% Calculamos el MSE
%Sumamos el error cuadrático de la imagen y dividimos por el número de elementos para
obtener el error cuadrático medio. Será un escalar.

%Tras aplicar al plano de intensidad
mseG = sum(sum(squaredErrorImageG)) / (rows * columns); % MSE entre la imagen
original y con ruido gaussiano
mseS = sum(sum(squaredErrorImageS)) / (rows * columns); % MSE entre la imagen
original y con ruido sal y pimienta

mseGf = sum(sum(squaredErrorImageGf)) / (rows * columns); % MSE entre la imagen
original y filtrada del ruido gaussiano
mseSf = sum(sum(squaredErrorImageSf)) / (rows * columns); % MSE entre la imagen
original y filtrada del ruido sal y pimienta

%A color
msecolorG = sum(sum(squaredErrorImagecolorG)) / (rows * columns); % MSE entre la
imagen original en color y con ruido gaussiano
msecolorS = sum(sum(squaredErrorImagecolorS)) / (rows * columns); % MSE entre la
imagen original en color y con ruido sal y pimienta

msecolorGf = sum(sum(squaredErrorImagecolorGf)) / (rows * columns); % MSE entre la
imagen original en color y filtrada del ruido gaussiano
msecolorSf = sum(sum(squaredErrorImagecolorSf)) / (rows * columns); % MSE entre la
imagen original en color y filtrada del ruido sal y pimienta

%% Calculamos la PSNR (Peak Signal to Noise Ratio) a partir del MSE de acuerdo con la
fórmula.

%Como estamos comparando la imagen original con la filtrada en el plano L, y
convertimos en RGB, ponemos 255.
PSNRg = 10 * log10( 255^2 / mseG); %De imagen original y ruido gaussiano
PSNRs = 10 * log10( 255^2 / mseS); %De imagen original y ruido sal y pimienta

PSNRgf = 10 * log10( 255^2 / mseGf);%De imagen original y filtrada de ruido gaussiano
PSNRsf = 10 * log10( 255^2 / mseSf);%De imagen original y filtrada de ruido sal y
pimienta

%A color, en el espacio RGB si va de 0 a 256
PSNRcolorg = 10 * log10( 255^2 / msecolorG); % De imagen original en color y ruido
gaussiano
PSNRcolors = 10 * log10( 255^2 / msecolorS); % De imagen original en color y ruido
sal y pimienta

PSNRcolorgf = 10 * log10( 255^2 / msecolorGf); % De imagen original en color y
filtrada de ruido gaussiano
PSNRcolorsf = 10 * log10( 255^2 / msecolorSf); % De imagen original en color y
filtrada de ruido sal y pimienta

% Alertas con los resultados.
%Tras aplicar al plano de intensidad
```

```
message1 = sprintf('\nMSE entre la imagen original y con ruido gaussiano = %.4f.\nThe
PSNR = %.4f', mseG, PSNRg);
message2 = sprintf('\nMSE entre la imagen original y con ruido sal y pimienta =
%.4f.\nThe PSNR = %.4f', mseS, PSNRs);
message3 = sprintf('\nMSE entre la imagen original y filtrada del ruido gaussiano =
%.4f.\nThe PSNR = %.4f', mseGf, PSNRgf);
message4 = sprintf('\nMSE entre la imagen original y filtrada del ruido sal y
pimienta = %.4f.\nThe PSNR = %.4f', mseSf, PSNRsf);
%A color
message5 = sprintf('\nMSE entre la imagen original en color y con ruido gaussiano =
%.4f.\nThe PSNR = %.4f', msecolorG, PSNRcolorg);
message6 = sprintf('\nMSE entre la imagen original en color y con ruido sal y
pimienta = %.4f.\nThe PSNR = %.4f', msecolorS, PSNRcolors);
message7 = sprintf('\nMSE entre la imagen original en color y filtrada del ruido
gaussiano = %.4f.\nThe PSNR = %.4f', msecolorGf, PSNRcolorgf);
message8 = sprintf('\nMSE entre la imagen original en color y filtrada del ruido sal
y pimienta = %.4f.\nThe PSNR = %.4f', msecolorSf, PSNRcolorsf);

%Representación en cajas
msgbox(message1,'Original/Ruido gaussiano');
msgbox(message2,'Original/Ruido sal y pimienta');

msgbox(message3,'Original/Fmedia gaussiano');
msgbox(message4,'Original/Fmedia sal y pimienta');

msgbox(message5,'Original/Ruido gaussiano espacio RGB');
msgbox(message6,'Original/Ruido sal y pimienta espacio RGB');

msgbox(message7,'Original/Fmedia gaussiano espacio RGB');
msgbox(message8,'Original/Fmedia sal y pimienta espacio RGB');
```

- Discusión de los resultados

De forma general, observamos que tras convertir al espacio CIE Lab y aplicar sólo la máscara al plano L, y además dependiendo de la cantidad de píxeles con ruido que haya en la imagen y de sus características propias, el filtro de media elimina gran parte de ese ruido, mejor con ruido gaussiano.

Por otro lado, al aplicar el filtro de media a cada plano R, G y B en lugar de uno solo, se reduce bastante más el error entre la imagen original y la imagen filtrada, por lo que la MSE es menor y la PSNR mayor.

4.6.2 Con filtro de mediana

Al igual que en el apartado 4.6.1, vamos a representar ahora los resultados de las medidas de fidelidad MSE y PSNR, en la Tabla 13 y Tabla 14, entre la imagen original y la misma imagen pero con ruido gaussiano o de sal y pimienta, y entre la imagen original y la imagen filtrada con el filtro de mediana. Tanto para el plano de intensidad L como en el espacio RGB.

Para la imagen en color aplicamos el filtro en cada plano, tal como anuncia el método de ordenamiento marginal.

Tabla 13. MSE y PSNR. Filtro de mediana, plano L.

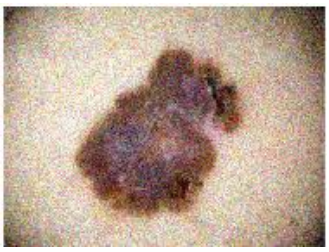
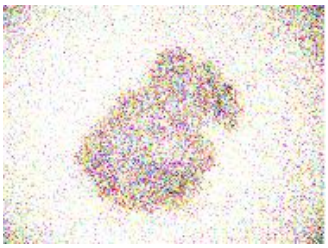
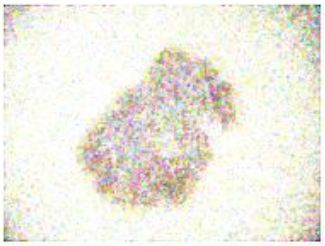
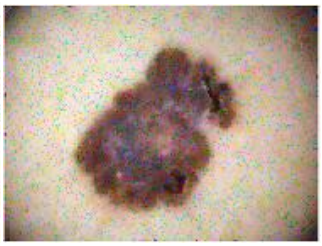
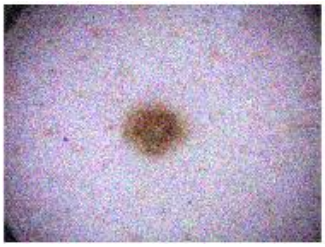
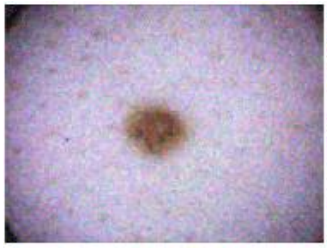
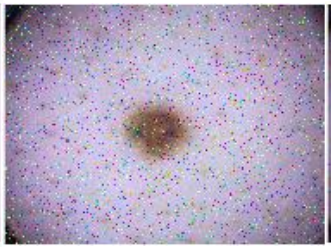
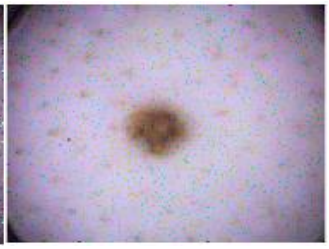
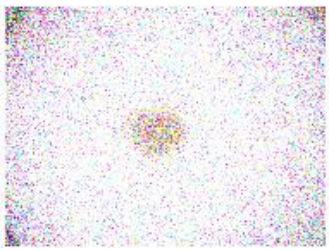
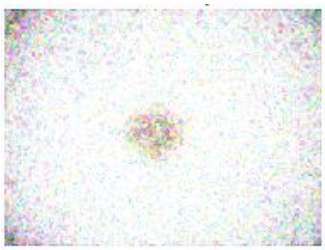
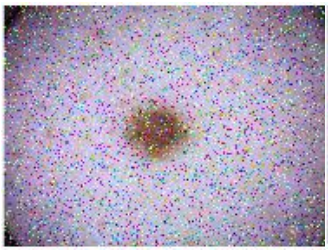
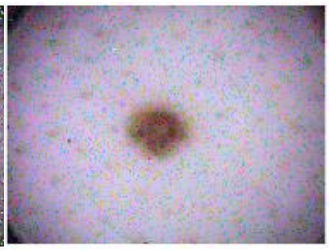
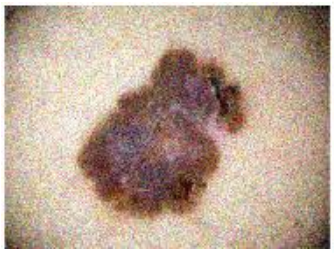
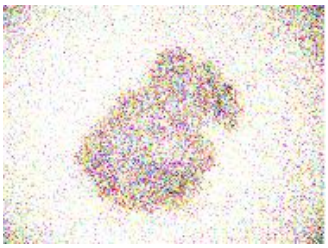
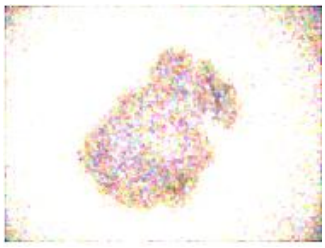
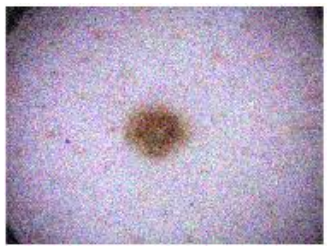
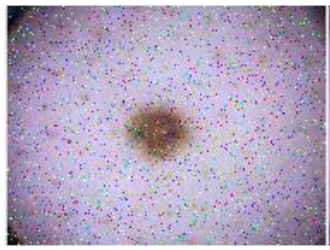
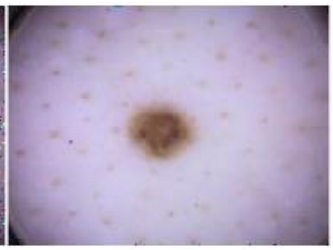
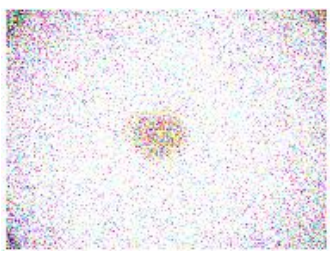
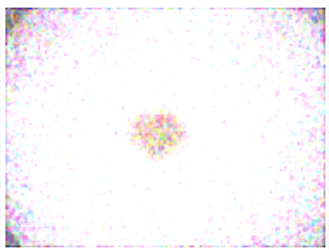
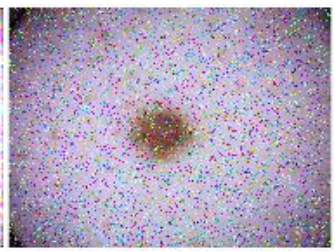
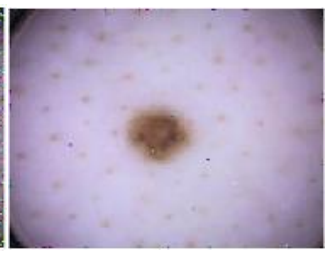
Imagen con ruido gaussiano	Imagen con filtro de mediana al ruido gaussiano	Imagen con ruido de sal y pimienta	Imagen con filtro de mediana al ruido sal y pimienta
			
media = 0 ; varianza = 0,01 mse = 0,0285 psnr = 63,5762	mse = 0,0260 psnr = 63,9787	densidad = 0,05 mse = 0,0475 psnr = 61,3683	mse = 0,0292 psnr = 63,4754
			
media = 0,5 ; varianza = 0,1 mse = 0,3312 psnr = 52,9295	mse = 0,3226 psnr = 53,0446	densidad = 0,1 mse = 0,0933 psnr = 58,4310	mse = 0,0564 psnr = 60,6195
			
media = 0 ; varianza = 0,01 mse = 0,0290 psnr = 63,5012	mse = 0,0256 psnr = 64,0451	densidad = 0,05 mse = 0,0458 psnr = 61,5216	mse = 0,0289 psnr = 63,5174
			
media = 0,5 ; varianza = 0,1 mse = 0,2555 psnr = 54,0569	mse = 0,2482 psnr = 54,1821	densidad = 0,1 mse = 0,0930 psnr = 58,4461	mse = 0,0580 psnr = 60,4930

Tabla 14. MSE y PSNR. Filtro de mediana, en RGB.

Imagen con ruido gaussiano	Imagen con filtro de media al ruido gaussiano	Imagen con ruido de sal y pimienta	Imagen con filtro de media al ruido sal y pimienta
			
media = 0 ; varianza = 0,01 mse = 0,0285 psnr = 63,5762	mse = 0,0056 psnr = 70,6347	densidad = 0,05 mse = 0,0475 psnr = 61,3683	mse = 0,0005 psnr = 81,0822
			
media = 0,5 ; varianza = 0,1 mse = 0,3312 psnr = 52,9295	mse = 0,3579 psnr = 52,5929	densidad = 0,1 mse = 0,0933 psnr = 58,4310	mse = 0,0012 psnr = 77,5211
			
media = 0 ; varianza = 0,01 mse = 0,0290 psnr = 63,5012	mse = 0,0053 psnr = 70,8567	densidad = 0,05 mse = 0,0458 psnr = 61,5216	mse = 0,0002 psnr = 84,6893
			
media = 0,5 ; varianza = 0,1 mse = 0,2555 psnr = 54,0569	mse = 0,2855 psnr = 53,5742	densidad = 0,1 mse = 0,0930 psnr = 58,4461	mse = 0,0007 psnr = 79,9909

Programado con:

```
i1 = 'ISIC_0498990.jpg';
i5 = 'ISIC_6821316.jpg';
f=imread(i1);
f=im2double(f); %Normalizamos los valores entre 0 y 1
%Añadimos ruido blanco gaussiano de media y varianza indicada por parámetros
fng=imnoise(f,"gaussian",0.5,0.1);
%Añadimos ruido impulsivo o salt & pepper, d=densidad %
fni=imnoise(f,"salt & pepper",0.1);

%Aplicamos el filtro de mediana a cada plano (se conoce como ordenamiento marginal)
sobre la imagen con ruido gaussiano.
fmedianaR_Gauss = medfilt2(fng(:,:,1));
fmedianaG_Gauss = medfilt2(fng(:,:,2));
fmedianaB_Gauss = medfilt2(fng(:,:,3));
fmedianaGauss = cat(3,fmedianaR_Gauss,fmedianaG_Gauss,fmedianaB_Gauss);

%Aplicamos el filtro de mediana a cada plano sobre la imagen con ruido impulsivo.
fmedianaR_Sal = medfilt2(fni(:,:,1));
fmedianaG_Sal = medfilt2(fni(:,:,2));
fmedianaB_Sal = medfilt2(fni(:,:,3));
fmedianaSal = cat(3,fmedianaR_Sal,fmedianaG_Sal,fmedianaB_Sal);

%Pintamos los resultados del filtrado al ruido
figure('Name','Filtro de mediana a imagen con ruido')
subplot(3,3,[1,4]),imshow(f); title('Imagen original')
subplot(3,3,2),imshow(fng);title('Imagen con ruido gaussiano')
subplot(3,3,3),imshow(fni);title('Imagen con ruido sal y pimienta')
subplot(3,3,5),imshow(fmedianaGauss);title('Imagen filtrada de ruido gaussiano')
subplot(3,3,6),imshow(fmedianaSal);title('Imagen filtrada de ruido sal y pimienta')

%% Convertimos la imagen de RGB a CIE.
% Imagen Original
fCIE=rgb2lab(f);  % Al cambiar del espacio rgb a lab, cambia el rango dinámico de la
imagen entre 0 y 100, por eso después lo representamos entre esos valores.
fCIEL=fCIE(:,:,1); % Obtenemos el plano L (escala de grises)
% Imagen con ruido GAUSSIANO
fngCIE=rgb2lab(fng);
fngCIEL=fngCIE(:,:,1);   % Obtenemos el plano L (escala de grises)
% Imagen con ruido SAL Y PIMIENTA
fniCIE=rgb2lab(fni);
fniCIEL=fniCIE(:,:,1);   % Obtenemos el plano L (escala de grises)

% Imagen filtrada de ruido gaussiano (filtro de mediana) al plano L
fmedianaCIELG=medfilt2(fngCIEL);
fmedianaCIEGf=cat(3,fmedianaCIELG,fngCIE(:,:,2),fngCIE(:,:,3));%Encadenamos los
planos
fmedianaCIEGf_RGB=lab2rgb(fmedianaCIEGf); %Pasamos a RGB
fmedianaCIEGf_RGB(fmedianaCIEGf_RGB > 1.0 ) = 1; %Umbralizamos los valores max y min
a 0 y 1
fmedianaCIEGf_RGB(fmedianaCIEGf_RGB < 0 ) = 0;

% Imagen filtrada de ruido sal y pimienta (filtro de mediana) al plano L
fmedianaCIELS=medfilt2(fniCIEL);
fmedianaCIESf=cat(3,fmedianaCIELS,fniCIE(:,:,2),fniCIE(:,:,3));%Encadenamos los
planos
fmedianaCIESf_RGB=lab2rgb(fmedianaCIESf); %Pasamos a RGB
fmedianaCIESf_RGB(fmedianaCIESf_RGB > 1.0 ) = 1; %Umbralizamos los valores max y min
a 0 y 1
fmedianaCIESf_RGB(fmedianaCIESf_RGB < 0 ) = 0;
```

```
% Representación en RGB tras media en el plano L de Lab.
figure('Name','RGB To Lab 3 planos')
subplot(3,3,[1,4]),imshow(f,[]); title('Imagen original') %Obtenemos el plano L
subplot(3,3,2),imshow(fng,[]);title('Imagen con ruido gaussiano')
subplot(3,3,3),imshow(fni,[]);title('Imagen con ruido sal y pimienta')
subplot(3,3,5),imshow(fmedianaCIEGf_RGB,[]);title('Lab:Filtrado Gauss paso a RGB')
subplot(3,3,6),imshow(fmedianaCIESf_RGB,[]);title('Lab:Filtrado S&P paso a RGB')

%----------------------- MEDIDADES DE FIDELIDAD -----------------------------------
% Vamos calcular la MSE y la PSNR como medidas de fidelidad de las imágenes

%Calculamos el error cuadrático en el espacio RGB tras solo aplicar la transformacion
al plano L
squaredErrorImageG = double((f(:,:,1)-fng(:,:,1)) .^ 2+(f(:,:,2)-fng(:,:,2)) .^
2+(f(:,:,3)-fng(:,:,3)) .^ 2);   %Entre la imagen original y con ruido gaussiano
squaredErrorImageS = double((f(:,:,1)-fni(:,:,1)) .^ 2+(f(:,:,2)-fni(:,:,2)) .^
2+(f(:,:,3)-fni(:,:,3)) .^ 2);   %Entre la imagen original y con ruido sal y pimienta

squaredErrorImageGf = double((f(:,:,1)-fmedianaCIEGf_RGB(:,:,1)) .^ 2+(f(:,:,2)-
fmedianaCIEGf_RGB(:,:,2)) .^ 2+(f(:,:,3)-fmedianaCIEGf_RGB(:,:,3)) .^ 2);    %Entre
la imagen original y filtrada del ruido gaussiano
squaredErrorImageSf = double((f(:,:,1)-fmedianaCIESf_RGB(:,:,1)) .^ 2+(f(:,:,2)-
fmedianaCIESf_RGB(:,:,2)) .^ 2+(f(:,:,3)-fmedianaCIESf_RGB(:,:,3)) .^ 2);    %Entre
la imagen original y filtrada del ruido sal y pimienta

%Error cuadrático en el espacio RGB
squaredErrorImagecolorG = double((f(:,:,1)-fng(:,:,1)) .^ 2+(f(:,:,2)-fng(:,:,2)) .^
2+(f(:,:,3)-fng(:,:,3)) .^ 2);  %Entre la imagen original y con ruido gaussiano.
squaredErrorImagecolorS = double((f(:,:,1)-fni(:,:,1)) .^ 2+(f(:,:,2)-fni(:,:,2)) .^
2+(f(:,:,3)-fni(:,:,3)) .^ 2);  %Entre la imagen original y con ruido sal y pimienta.

squaredErrorImagecolorGf = double((f(:,:,1)-fmedianaGauss(:,:,1)) .^ 2+(f(:,:,2)-
fmedianaGauss(:,:,2)) .^ 2+(f(:,:,3)-fmedianaGauss(:,:,3)) .^ 2); %Entre la imagen
original y filtrada del ruido gaussiano.
squaredErrorImagecolorSf = double((f(:,:,1)-fmedianaSal(:,:,1)) .^ 2+(f(:,:,2)-
fmedianaSal(:,:,2)) .^ 2+(f(:,:,3)-fmedianaSal(:,:,3)) .^ 2); %Entre la imagen
original y filtrada del ruido sal y pimienta.

% Obtenemos los píxeles de la imagen.
[rows columns plane] = size(f); %Dividimos en filas, columnas y planos.

%% Calculamos el MSE. Sumamos el error cuadrático de la imagen y dividimos por el
número de elementos para obtener el error cuadrático medio. Será un escalar.
%Tras aplicar al plano de intensidad
mseG = sum(sum(squaredErrorImageG)) / (rows * columns); % MSE entre la imagen
original y con ruido gaussiano.
mseS = sum(sum(squaredErrorImageS)) / (rows * columns); % MSE entre la imagen
original y con ruido sal y pimienta.

mseGf = sum(sum(squaredErrorImageGf)) / (rows * columns); % MSE entre la imagen
original y filtrada del ruido gaussiano
mseSf = sum(sum(squaredErrorImageSf)) / (rows * columns); % MSE entre la imagen
original y filtrada del ruido sal y pimienta

%A color
msecolorG = sum(sum(squaredErrorImagecolorG)) / (rows * columns); % MSE entre la
imagen original en color y con ruido gaussiano
msecolorS = sum(sum(squaredErrorImagecolorS)) / (rows * columns); % MSE entre la
imagen original en color y con ruido sal y pimienta
```

```
msecolorGf = sum(sum(squaredErrorImagecolorGf)) / (rows * columns); % MSE entre la
imagen original en color y filtrada del ruido gaussiano
msecolorSf = sum(sum(squaredErrorImagecolorSf)) / (rows * columns); % MSE entre la
imagen original en color y filtrada del ruido sal y pimienta

% Calculamos la PSNR (Peak Signal to Noise Ratio) a partir del MSE de acuerdo con la
fórmula.
%Como estamos comparando la imagen original con la filtrada en el plano L, y
convertimos en RGB, ponemos 255.
PSNRg = 10 * log10( 255^2 / mseG); % De imagen original y ruido gaussiano
PSNRs = 10 * log10( 255^2 / mseS); % De imagen original y ruido sal y pimienta

PSNRgf = 10 * log10( 255^2 / mseGf);% De imagen original y filtrada de ruido
gaussiano
PSNRsf = 10 * log10( 255^2 / mseSf);% De imagen original y filtrada de ruido sal y
pimienta

%A color, en el espacio RGB si va de 0 a 256
PSNRcolorg = 10 * log10( 255^2 / msecolorG);%De imagen original en color y ruido
gaussiano
PSNRcolors = 10 * log10( 255^2 / msecolorS);%De imagen original en color y ruido sal
y pimienta

PSNRcolorgf = 10 * log10( 255^2 / msecolorGf);%De imagen original en color y filtrada
de ruido gaussiano
PSNRcolorsf = 10 * log10( 255^2 / msecolorSf);%De imagen original en color y filtrada
de ruido sal y pimienta

% Alertas con los resultados.
%En el plano L
message1 = sprintf('\nMSE entre la imagen original y con ruido gaussiano = %.4f.\nThe
PSNR = %.4f', mseG, PSNRg);
message2 = sprintf('\nMSE entre la imagen original y con ruido sal y pimienta =
%.4f.\nThe PSNR = %.4f', mseS, PSNRs);
message3 = sprintf('\nMSE entre la imagen original y filtrada del ruido gaussiano =
%.4f.\nThe PSNR = %.4f', mseGf, PSNRgf);
message4 = sprintf('\nMSE entre la imagen original y filtrada del ruido sal y
pimienta = %.4f.\nThe PSNR = %.4f', mseSf, PSNRsf);
%A color
message5 = sprintf('\nMSE entre la imagen original en color y con ruido gaussiano =
%.4f.\nThe PSNR = %.4f', msecolorG, PSNRcolorg);
message6 = sprintf('\nMSE entre la imagen original en color y con ruido sal y
pimienta = %.4f.\nThe PSNR = %.4f', msecolorS, PSNRcolors);
message7 = sprintf('\nMSE entre la imagen original en color y filtrada del ruido
gaussiano = %.4f.\nThe PSNR = %.4f', msecolorGf, PSNRcolorgf);
message8 = sprintf('\nMSE entre la imagen original en color y filtrada del ruido sal
y pimienta = %.4f.\nThe PSNR = %.4f', msecolorSf, PSNRcolorsf);

%Representación en cajas
msgbox(message1,'Original/Ruido gaussiano');
msgbox(message2,'Original/Ruido sal y pimienta');

msgbox(message3,'Original/Fmediana gaussiano');
msgbox(message4,'Original/Fmediana sal y pimienta');

msgbox(message5,'Original/Ruido gaussiano espacio RGB');
msgbox(message6,'Original/Ruido sal y pimienta espacio RGB');

msgbox(message7,'Original/Fmediana gaussiano espacio RGB');
msgbox(message8,'Original/Fmediana sal y pimienta espacio RGB');
```

- Discusión de los resultados

Destaca sobre todo como el filtro de mediana elimina mucho el ruido de sal y pimienta, más al aplicar el filtro a los tres planos de color RGB que al aplicarlo a uno solo, en el caso del plano intensidad L del espacio CIE Lab.

Por otro lado, vemos que si se aumenta los valores de media y varianza del ruido gaussiano lo suficiente, el filtro de media en lugar de mejorar la imagen, la empeora, existiendo más error en comparación con la imagen original.

Como queríamos demostrar, el filtro de mediana funciona mucho mejor de forma general en imágenes con ruido de sal y pimienta que en imágenes con ruido gaussiano.

4.7 Implementación de medidas no referenciadas

Tal y como hemos definido en el capítulo 2, las medidas no referenciadas se aplican sobre una imagen digital, sin disponer de la imagen original para tomarla como referencia y comparar así los valores.

Vamos a aplicar entonces, los métodos descritos en el apartado 3.3, ecualización del histograma, enmascaramiento del desenfoque y filtrado de refuerzo de las frecuencias altas y comparamos de esta forma, los valores del contraste, la croma y la nitidez de la imagen antes de la mejora y después.

4.7.1 Para el contraste

Ya que el contraste, como ya hemos definido en el capítulo 2, es el nivel o relación de intensidad de cada uno de los puntos que forman la imagen. Vamos a aplicar las transformaciones necesarias en los planos de intensidad del espacio de color que corresponda, en nuestro caso en el plano V de HSV.

Su implementación en MATLAB se ha llevado a cabo a través del siguiente código:

```
i2 = 'ISIC_0984817.jpg';
i4 = 'ISIC_4602463.jpg';
i6 = 'ISIC_6948176.jpg';
f=imread(i6);
%Tranformamos la imagen a doble y normalizada
fdouble= im2double(f);
fHSV= rgb2hsv(fdouble);

%% Representación gráfica de las imágenes originales
figure('Name','Imágenes originales')
subplot(3,3,[1,4]),imshow(f); title('Imagen original')
subplot(3,3,2),imshow(fdouble);title('Imagen Double')
subplot(3,3,3),imshow(fHSV);title('Imagen HSV')
subplot(3,3,5),imshow(fdouble(:,:,3));title('Imagen Double Plano 3')
subplot(3,3,6),imshow(fHSV(:,:,3));title('Plano V de HSV')

%% Cálculo del contraste por parecido cercano entre píxeles de la imagen original
Adouble=fdouble(:,:,3);%Plano 3 de imagen en double
AI=fHSV(:,:,3);        %Plano 3 de imagen en HSV
visionkD = calculo_contraste(Adouble)
visionkI = calculo_contraste(AI)
```

```
%% Mejora del contraste con Ecualizacion del histograma.
feV=histeq(fHSV(:,:,3));%Calculamos la ecualización del histograma de la componente V
fefinV=cat(3,fHSV(:,:,1),fHSV(:,:,2),feV); %Encademos todos los planos
feHSV_RGB=hsv2rgb(fefinV); %Convertimos a RGB
%Para el cálculo del contraste tomamos el plano V de la imagen ecualizada en HSV
AE=fefinV(:,:,3);
visionkE = calculo_contraste(AE)

%% Representación gráfica de las imágenes tras la ecualización del histograma
figure('Name','Ecualización del histograma')
subplot(1,3,1),imshow(f);title('Imagen original')
subplot(1,3,2),imshow(fefinV);title('Imagen ecualizada HSV')
subplot(1,3,3),imshow(feHSV_RGB);title('Imagen ecualizada RGB')

%% Contraste con USM
%Definimos la máscara y aplicamos su filtrado con filter2() a la componente V
maskUSM=[-1/8 -1/8 -1/8;-1/8 2 -1/8;-1/8 -1/8 -1/8];
gfilUSM=filter2(maskUSM,fHSV(:,:,3));
gfilUSM(gfilUSM > 1.0 ) = 1; %Umbralizamos los valores max y min a 0 y 1
gfilUSM(gfilUSM < 0 ) = 0;

%Encadenamos la componete V a la imagen
gfinUSM=cat(3,fHSV(:,:,1),fHSV(:,:,2),gfilUSM);
%Transformamos a RGB
fUSM_RGB=hsv2rgb(gfinUSM);

figure('Name','USM')
subplot(2,2,1),imshow(fdouble);title('Imagen Original')
subplot(2,2,2),imshow(gfilUSM,[]);title('Imagen filtrada, solo V')
subplot(2,2,3),imshow(gfinUSM,[]);title('Imagen encadenada HSV')
subplot(2,2,4),imshow(fUSM_RGB,[]);title('Imagen USM en RGB')

%Para el cálculo del contraste tomamos el plano V de la imagen USM en HSV
AUSM=gfinUSM(:,:,3);
visionkUSM = calculo_contraste(AUSM)

%% Contraste con High Boost filtering
%Definimos la máscara y aplicamos su filtrado con filter2() a la componente V
maskHBF=[-1 -1 -1;-1 9 -1;-1 -1 -1];
gfilHB=filter2(maskHBF,fHSV(:,:,3));
gfilHB(gfilHB > 1.0 ) = 1;
gfilHB(gfilHB < 0 ) = 0;

gfinHB=cat(3,fHSV(:,:,1),fHSV(:,:,2),gfilHB);%Encadenamos la componente V a la imagen
fHB_RGB=hsv2rgb(gfinHB); %Transformamos a RGB

figure('Name','HB')
subplot(2,2,1),imshow(fdouble);title('Imagen Original')
subplot(2,2,2),imshow(gfilHB,[]);title('Imagen filtrada,solo V')
subplot(2,2,3),imshow(gfinHB,[]);title('Imagen encadenada HSV')
subplot(2,2,4),imshow(fHB_RGB,[]);title('Imagen HBF RGB')

%Para el cálculo del contraste tomamos el plano V de la imagen USM en HSV
AHB=gfinHB(:,:,3);
visionkHB = calculo_contraste(AHB)
```

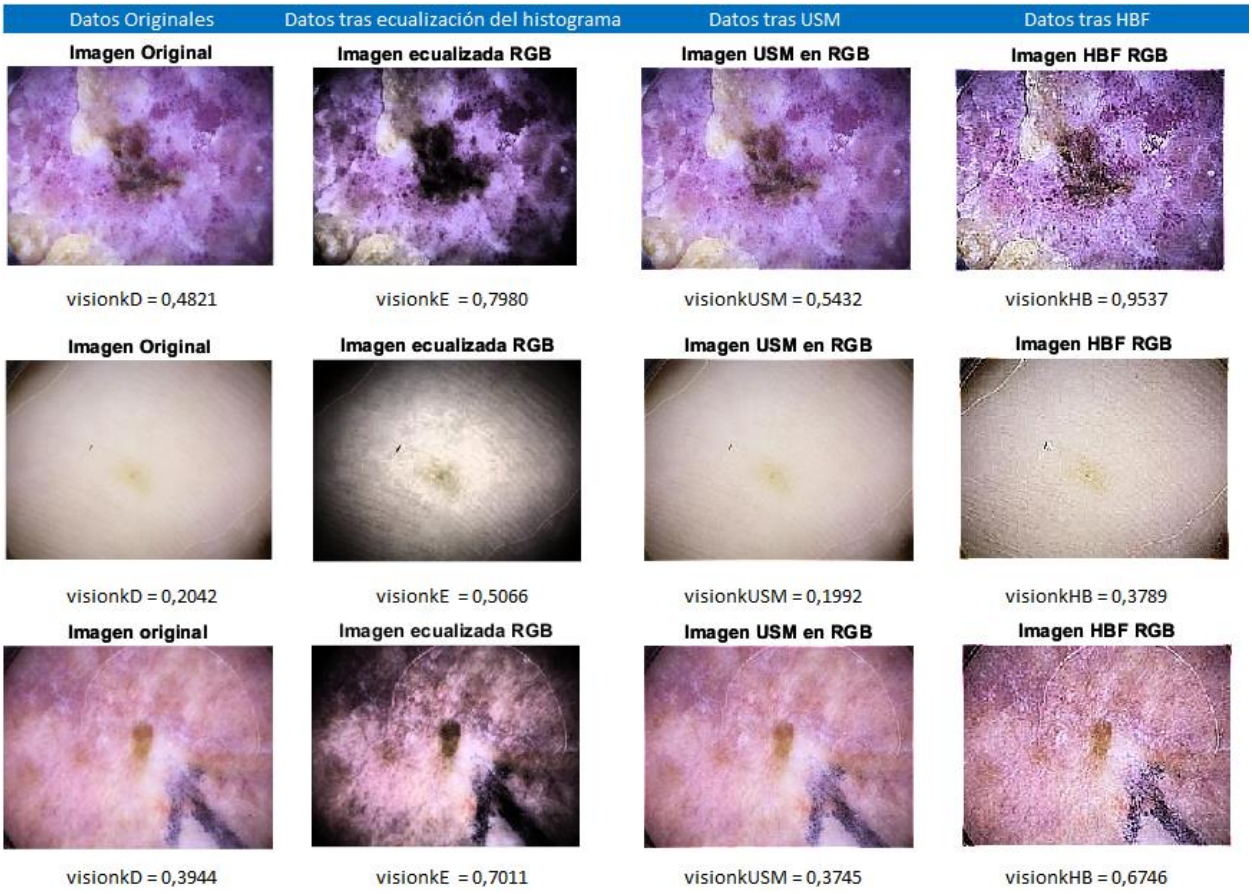
Definición de la función `calculo_contraste()`:

```
function [visionk] = calculo_contraste(A)
%Función que calcula el contraste de una parte de la imagen a través del "parecido"
entre píxeles cercanos.
[filas,columnas]=size(A);
```

```
clear vision;
vision=[];
k=1;
Recorremos un área rectangular de la imagen de 11x36
for i=1:10:(filas-11)
for j=1:10:(columnas-36)
m1=max(max(A(i:i+10,j:j+35)));  %Tomamos los máx y min de esas áreas
m2=min(min(A(i:i+10,j:j+35)));
vision(k)=(m1-m2)/(m1);         %Fórmula del contraste de Michelson
k=k+1;
end
end
%El valor del contraste será la media de todos los valores obtenidos al recorrer el
área delimitada.
visionk=mean(vision);
end
```

Los resultados se pueden observar en la Tabla 15:

Tabla 15. Medidas del contraste.



- Discusión de los resultados

Los valores están normalizados entre 0 y 1.
De forma general, observamos que la ecualización del histograma mejora bastante el contraste de la imagen original, la ecualización será mejor cuanto más oscura sea la imagen o menos uniforme sean los valores. Aunque como podemos observar al aplicar la ecualización del histograma a una imagen en color se deforman los colores originales.

El método USM, en la mayoría de los casos, no lo mejora nada, es más, lo empeora, ya que tanto este método como HBF mejoran sobre todo la nitidez de los bordes.

El filtro de refuerzo de las frecuencias altas vemos que mejora el contraste en algunos casos, pero arroja resultados engañosos según las características de las imágenes originales, ya que se encarga sobre todo de mejorar el emborronado.

4.7.2 Para la croma

Esta característica está presente como protagonista en algunos planos de color, como por ejemplo en el segundo plano del espacio LCH. Vamos a medir el colorido, que como describimos en el apartado 2.2.2, está relacionado con la croma y la saturación.

Desarrollo en MATLAB:

```
i2 = 'ISIC_0984817.jpg';
i4 = 'ISIC_4602463.jpg';
i6 = 'ISIC_6948176.jpg';
f=imread(i2);
fdouble= im2double(f); %Transformamos la imagen a doble y normalizada entre [0,1].
flab=rgb2lab(fdouble); %Transformamos a LAB.
cM=makecform('lab2lch'); %Aplicamos esta función para pasar a lch (donde está la
croma(c) y la saturación(h)). Crea la estructura del tamaño de un plano (192x256).
flch=applycform(flab,cM); %Aplica la transformación a LCH con la estructura cM sobre
flab.

%Separamos los planos LCH
l=flch(:,:,1); c=flch(:,:,2); h=flch(:,:,3);

%Máscara UNSHARP MASKING
maskUSM=[-1/8 -1/8 -1/8;-1/8 2 -1/8;-1/8 -1/8 -1/8];
%Máscara HIGH BOOST
maskHBF=[-1 -1 -1;-1 9 -1;-1 -1 -1];

gfilUSM=filter2(maskUSM,l); %Aplicamos las máscaras al plano intensidad L.
gfilHBF=filter2(maskHBF,l);
gfinUSM=cat(3,gfilUSM,c,h); %Encadenamos las 3 componentes a la imagen para los
métodos.
gfinHBF=cat(3,gfilHBF,c,h);

%Transformamos primero a Lab y después a RGB para visualizar los resultados.
cform=makecform('lch2lab');
gUSM_LAB=applycform(gfinUSM,cform);
gHBF_LAB=applycform(gfinHBF,cform);
gUSM_RGB=lab2rgb(gUSM_LAB);
gUSM_RGB(gUSM_RGB > 1.0 ) = 1; %Umbralizamos los valores max y min a 0 y 1
gUSM_RGB(gUSM_RGB < 0 ) = 0;
gHBF_RGB=lab2rgb(gHBF_LAB);
gHBF_RGB(gHBF_RGB > 1.0 ) = 1;
gHBF_RGB(gHBF_RGB < 0 ) = 0;

%% Medidas COLORFULNESS de imagen original
%colorFulnessOriginal = calcula_colorido(flch(:,:,2)) % ¿Hay q calcular el colorido
sólo de la componente C o de todos los planos?
colorFulnessOriginal = calcula_colorido(fdouble)

%% Medidas COLORFURNESS de imagen final (tras ecualización de histograma)
fHSV= rgb2hsv(fdouble);                    %Transformamos a HSV.
feV=histeq(fHSV(:,:,3));%Calculamos la ecualización del histograma de la componente V
```

```
fefinV=cat(3,fHSV(:,:,1),fHSV(:,:,2),feV); %Encadenamos los planos.
feHSV_RGB=hsv2rgb(fefinV);                 %Transformamos a RGB.

colorFulness_ecualiza = calcula_colorido(feHSV_RGB)

%% Medidas COLORFURNESS de imagen final (tras la máscara: Metodos USM y HighBoost)
colorFulness_USM = calcula_colorido(gUSM_RGB)
colorFulness_HBF = calcula_colorido(gHBF_RGB)

%% Representación gráfica
figure('Name','Imagenes')
subplot(3,3,1),imshow(f);title('Imagen Original')
subplot(3,3,2),imshow(fdouble);title('Imagen Original Double')
subplot(3,3,4),imshow(l,[]);title('Plano L')
subplot(3,3,5),imshow(c,[]);title('Plano C')
subplot(3,3,6),imshow(h,[]);title('Plano H')
subplot(3,3,7),imshow(feHSV_RGB);title('Imagen ecualizada')
subplot(3,3,8),imshow(gUSM_RGB,[]);title('Imagen USM en RGB')
subplot(3,3,9),imshow(gHBF_RGB,[]);title('Imagen HBF en RGB')
```

Función `calcula_colorido()`:

```
function [colorFulness] = calcula_colorido(fimg)
%Función que calcula el colorido (colorfulness) a partir de la media y varianza
 m=mean2(fimg); %mean2 hace la media de la imagen entera (no por filas y columnas
como hace mean)
 var=std2(fimg); %es la variación típica (la varianza es la variación tipica^2)
 colorFulness = m + var;
end
```

Resultados en la Tabla 16.

Tabla 16. Medidas del colorido.

Datos Originales	Datos tras ecualización del histograma	Datos tras USM	Datos tras HBF
Imagen Original	**Imagen ecualizada**	**Imagen USM en RGB**	**Imagen HBF en RGB**
			
colorFulnessOriginal = 0,7848	colorFulness_ecualiza = 0,6936	colorFulness_USM = 0,7928	colorFulness_HBF = 0,8567
Imagen Original	**Imagen ecualizada**	**Imagen USM en RGB**	**Imagen HBF en RGB**
			
colorFulnessOriginal = 0,8478	colorFulness_ecualiza = 0,7514	colorFulness_USM = 0,8505	colorFulness_HBF = 0,8626
Imagen Original	**Imagen ecualizada**	**Imagen USM en RGB**	**Imagen HBF en RGB**
			
colorFulnessOriginal = 0,8043	colorFulness_ecualiza = 0,7114	colorFulness_USM = 0,8099	colorFulness_HBF = 0,8526

- Discusión de los resultados

Vemos que, en todos los casos, al aplicar la ecualización del histograma, el colorido empeora. Sin embargo, con los métodos USM y HBF hay una ligera mejora, dependiendo de las características de cada imagen. Tiene sentido ya que ninguno de estos métodos se centran en mejorar el colorido de una imagen.

4.7.3 Para la nitidez

En el capítulo 2 describimos esta característica como el emborronado que poseen los bordes de los elementos de las imágenes digitales. Para ello usamos el gradiente de cada plano que la componen.

Vamos a ver cómo se comporta cada método con esta medida.

Desarrollo en MATLAB:

```
i2 = 'ISIC_0984817.jpg';
i4 = 'ISIC_4602463.jpg';
i6 = 'ISIC_6948176.jpg';
f=imread(i6);
fdouble= im2double(f);    %Transformamos la imagen a doble y normalizada.
%Separamos los planos RGB.
r=fdouble(:,:,1); g=fdouble(:,:,2); b=fdouble(:,:,3);

%% Obtenemos el valor de sharpness de la imagen inicial de cada plano para calcularle
el gradiente.
sharpness_originalR=estimate_sharpness(fdouble(:,:,1));
sharpness_originalG=estimate_sharpness(fdouble(:,:,2));
sharpness_originalB=estimate_sharpness(fdouble(:,:,3));
sharpness_original=sharpness_originalR + sharpness_originalG + sharpness_originalB

%% Medidas de sharpness tras ecualización de histograma
fHSV= rgb2hsv(fdouble); %Convertimos al espacio HSV
feV=histeq(fHSV(:,:,3));%Calculamos la ecualización del histograma de la componente V
fefinV=cat(3,fHSV(:,:,1),fHSV(:,:,2),feV);
feHSV_RGB=hsv2rgb(fefinV);

sharpness_ecualizaR=estimate_sharpness(feHSV_RGB(:,:,1));
sharpness_ecualizaG=estimate_sharpness(feHSV_RGB(:,:,2));
sharpness_ecualizaB=estimate_sharpness(feHSV_RGB(:,:,3));
sharpness_ecualiza=sharpness_ecualizaR + sharpness_ecualizaG + sharpness_ecualizaB

%Máscara UNSHARP MASKING
maskUSM=[-1/8 -1/8 -1/8;-1/8 2 -1/8;-1/8 -1/8 -1/8];
%Máscara HIGH BOOST
maskHBF=[-1 -1 -1;-1 9 -1;-1 -1 -1];

%% Filtramos con el método USM a cada componente RGB.
gUSM_R=filter2(maskUSM,r);
gUSM_G=filter2(maskUSM,g);
gUSM_B=filter2(maskUSM,b);
gUSMfin=cat(3,gUSM_R,gUSM_G,gUSM_B); %Encadenamos las 3 componentes a la imagen.
gUSMfin(gUSMfin > 1.0 ) = 1;
gUSMfin(gUSMfin < 0 ) = 0;

% Sharpness de la imagen tratada con USM.
sharpness_usm3R=estimate_sharpness(gUSMfin(:,:,1));
sharpness_usm3G=estimate_sharpness(gUSMfin(:,:,2));
```

```
sharpness_usm3B=estimate_sharpness(gUSMfin(:,:,3));
sharpness_usm3=sharpness_usm3R + sharpness_usm3G + sharpness_usm3B

%% Filtramos con el método USM en el plano V
gUSM_V=filter2(maskUSM,fHSV(:,:,3));
gUSM_Vfin=cat(3,fHSV(:,:,1),fHSV(:,:,2),gUSM_V); %Encadenamos las 3 componentes
gUSM_HSV_RGB=hsv2rgb(gUSM_Vfin);
gUSM_HSV_RGB(gUSM_HSV_RGB > 1.0 ) = 1; %Umbralizamos los valores max y min a 0 y 1
gUSM_HSV_RGB(gUSM_HSV_RGB < 0 ) = 0;

% Sharpness de la imagen tratada con USM.
sharpness_usmR=estimate_sharpness(gUSM_HSV_RGB(:,:,1));
sharpness_usmG=estimate_sharpness(gUSM_HSV_RGB(:,:,2));
sharpness_usmB=estimate_sharpness(gUSM_HSV_RGB(:,:,3));
sharpness_usm=sharpness_usmR + sharpness_usmG + sharpness_usmB

%% Filtramos con el método HBF a cada componente RGB
gHBF_R=filter2(maskHBF,r);
gHBF_G=filter2(maskHBF,g);
gHBF_B=filter2(maskHBF,b);
gHBFfin=cat(3,gHBF_R,gHBF_G,gHBF_B); %Encadenamos las 3 componentes a la imagen
gHBFfin(gHBFfin > 1.0 ) = 1;
gHBFfin(gHBFfin < 0 ) = 0;

% Sharpness de la imagen tratada con HBF
sharpness_hbf3R=estimate_sharpness(gHBFfin(:,:,1));
sharpness_hbf3G=estimate_sharpness(gHBFfin(:,:,2));
sharpness_hbf3B=estimate_sharpness(gHBFfin(:,:,3));
sharpness_hbf3=sharpness_hbf3R + sharpness_hbf3G + sharpness_hbf3B

%% Filtramos con el método HBF en el plano V
gHBF_V=filter2(maskHBF,fHSV(:,:,3));
gHBF_Vfin=cat(3,fHSV(:,:,1),fHSV(:,:,2),gHBF_V);%Encadenamos las 3 componentes

gHBF_HSV_RGB=hsv2rgb(gHBF_Vfin);
gHBF_HSV_RGB(gHBF_HSV_RGB > 1.0 ) = 1;
gHBF_HSV_RGB(gHBF_HSV_RGB < 0 ) = 0;

% Sharpness de la imagen tratada con HBF
sharpness_hbfR=estimate_sharpness(gHBF_HSV_RGB(:,:,1));
sharpness_hbfG=estimate_sharpness(gHBF_HSV_RGB(:,:,2));
sharpness_hbfB=estimate_sharpness(gHBF_HSV_RGB(:,:,3));
sharpness_hbf=sharpness_hbfR + sharpness_hbfG + sharpness_hbfB

%% Representación gráfica
figure('Name','Imágenes')
subplot(3,2,1),imshow(f);title('Imagen Original')
subplot(3,2,2),imshow(feHSV_RGB);title('Imagen ecualizada')
subplot(3,2,3),imshow(gUSMfin,[]);title('Imagen filtrada USM en 3 planos')
subplot(3,2,4),imshow(gHBFfin,[]);title('Imagen filtrada HBF en 3 planos')
subplot(3,2,5),imshow(gUSM_HSV_RGB,[]);title('Imagen filtrada USM en V')
subplot(3,2,6),imshow(gHBF_HSV_RGB,[]);title('Imagen filtrada HBF en V')

%% Función que calcula la medida de sharpness a través del gradiente
function [sharpness]=estimate_sharpness(G)
[Gx, Gy]=gradient(G);
S=sqrt(Gx.*Gx+Gy.*Gy);
sharpness=sum(sum(S))./(numel(Gx)); %Promedio del módulo del gradiente.
End
```

Mostramos los resultados en la Tabla 17.

Tabla 17. Medidas de la nitidez.

Datos Originales	Datos tras ecualización del histograma	Datos tras USM	Datos tras HBF
Imagen Original  sharpness_original = 0,0972	Imagen ecualizada  sharpness_ecualiza = 0,1498	Imagen filtrada USM en 3 planos  sharpness_usm3 = 0,1516 Imagen filtrada USM en V  sharpness_usm = 0,1433	Imagen filtrada HBF en 3 planos  sharpness_hbf3 = 0,4911 Imagen filtrada HBF en V  sharpness_hbf = 0,4363
Imagen Original  sharpness_original = 0,0240	Imagen ecualizada  sharpness_ecualiza = 0,0530	Imagen filtrada USM en 3 planos  sharpness_usm3 = 0,0439 Imagen filtrada USM en V  sharpness_usm = 0,0430	Imagen filtrada HBF en 3 planos  sharpness_hbf3 = 0,1355 Imagen filtrada HBF en V  sharpness_hbf = 0,1286
Imagen Original  sharpness_original = 0,0630	Imagen ecualizada  sharpness_ecualiza = 0,1247	Imagen filtrada USM en 3 planos  sharpness_usm3 = 0,1005 Imagen filtrada USM en V  sharpness_usm = 0,0964	Imagen filtrada HBF en 3 planos  sharpness_hbf3 = 0,3267 Imagen filtrada HBF en V  sharpness_hbf = 0,2948

- Discusión de los resultados

Hemos aplicado las máscaras de USM y HBF en primer lugar a cada plano R, G y B, obteniendo las medidas *sharpness_usm3* y *sharpness_hbf;* y en segundo lugar sólo al plano de intensidad V del espacio HSV.

Tanto el método USM como HBF mejoran la nitidez de los bordes de la imagen, en menor medida el primero que el segundo, de forma general, al realzar las frecuencias altas.

Por tanto, el método HBF es el que arroja mejores resultados en la mejora de la nitidez y por tanto el emborronado de los bordes.

5 CONCLUSIONES

Para finalizar, podemos concluir que las medidas de calidad de la imagen, tanto referenciadas como no referenciadas, son de gran ayuda a la hora de comparar, analizar y discutir cuánto de "buena" puede ser una imagen, si existe mucho error entre los píxeles originales y los actuales, cómo es su contraste, el grado de intensidad que tiene en relación al blanco, su saturación, su nitidez en los bordes, etc.

Vamos a mostrar en la Tabla 18 las conclusiones generales para cada medidad de calidad.

Tabla 18. Conclusiones finales.

	MSE y PSNR	El contraste	La croma	La nitidez
Filtro de media	Se mejora la imagen en cuanto al error, sobre todo si posee un ruido gaussiano.			
Filtro de mediana	Funciona mucho mejor si la imagen tiene imperfecciones del tipo sal y pimienta.			
Ecualización del histograma		Es sin duda el mejor método de mejora para esta característica, ya que medimos el parecido de los píxeles solo en el plano de intensidad.	Se reduce al aplicar este método, ya que solo se actúa sobre el plano de intensidad para sumar la media y la desviación típica.	Puede mejorar un poco, dependiendo de la imagen pero la nitidez de los bordes se resaltan mejor con los métodos USM y HBF.
Unsharp masking		Es el método que peor funciona para el contraste, no existe mucha mejora.	No se ven apenas modificadas puesto que la mejora se aplica al plano intensidad.	Este método junto con HBF son generalmente los mejores para esta característica.
High Boost filtering		Mejora bastante el contraste de la imagen si ésta posee frecuencias altas, ya que filtramos con una máscara que las enfatiza.	Lo mismo sucede con este método, apenas hay variación.	Es el mejor método para mejorar el emborronado de los bordes y por tanto la nitidez.

REFERENCIAS

[1] Rangaraj M. Rangayyan, Begoña Acha, Carmen Serrano. Color image processing with biomedical applications, pages 117-118.

[2] Winkler S. Visual fidelity and perceived quality: towards comprehensive metrics. In Rogowitz BE and Pappas TN, editors, *Proceedings of SPIE*, volume 4299, pages 114-125, 2001. doi:10.1117/12.429540.

[3] A.N. Netravali and B.G. Haskell, Digital Pictures: Representation, Compression, and Standards (2nd Ed), Plenum Press, New York, NY (1995).

[4] Michelson, A. (1927). Studies in Optics. U. of Chicago Press.

[5] E. Peli (Oct 1990). "Contrast in Complex Images". Journal of the Optical Society of America A. 7 (10): 2032–2040.

[6] Fedorovskaya EA, Ridder H, and Blommaert FJJ. Chroma variations and perceived quality of color images of natural scenes. Color Research and Application, 22(2): 96-110,1997.

[7] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins. Digital Image Processing Using MATLAB, pages 320-321.

[8] Rafael C. Gonzalez, Richard E. Woods. Digital Image Processing, pages 320-323.

[9] Rangaraj M. Rangayyan, Begoña Acha, Carmen Serrano. Color image processing with biomedical applications, pages 175-178.

[10] Rangaraj M. Rangayyan, Begoña Acha, Carmen Serrano. Color image processing with biomedical applications, pages 215-216.

[11] Rafael C. Gonzalez, Richard E. Woods. Digital Image Processing, pages 134-140.

[12] Carl-Fredrik Westin Ron Kikinis, Hans Knutsson, in Handbook of Medical Imaging, 2000. Adaptive Image Filtering.

[13] Kamal Sharma, Akhil Gupta, Bandana Sharma, Suman Lata Tripathi. Intelligent Communication and Automation Systems, pages 70-72.

[14] Rangaraj M. Rangayyan, Begoña Acha, Carmen Serrano. Color image processing with biomedical applications, pages 229-233.