

# Procesamiento avanzado de imágenes astronómicas

Bruno Quadrelli

Franco Ribarov

Emmanouil-Angelos Tsigkas

*Procesamiento de Imágenes Digitales*

Universidad de Sevilla

13 de mayo de 2025

## 1. Resumen

En la presente memoria se aborda la implementación de algoritmos avanzados de procesamiento de imágenes digitales, particularmente aplicados en imágenes astronómicas para la detección de cuerpos celestes, basándonos en el artículo *Advanced Image Processing for Astronomical Images* de *Diganta Misra, Sparsha Mishra y Bhargav Appasani*. Es un proyecto desarrollado en Python, un lenguaje conocido y sumamente utilizado en el ámbito de procesamiento de imágenes. Una imagen para ser clasificada debe pasar por un procesamiento de la misma, luego será analizada para obtener nueva información y por último será la entrada para un modelo entrenado de Machine Learning para la clasificación de cuerpos celestes.

# Índice

<b>1. Resumen</b>	<b>2</b>
<b>2. Introducción</b>	<b>9</b>
<b>3. Objetivo</b>	<b>11</b>
3.1. Pipeline . . . . .	11
<b>4. Planteamiento teórico</b>	<b>13</b>
4.1. Eliminación de ruido . . . . .	13
4.1.1. Non-Local Means . . . . .	13
4.1.2. Power Spectrum Analysis . . . . .	14
4.2. Detección de Bordes . . . . .	17
4.2.1. Canny . . . . .	17
4.2.2. Laplaciana de Gauss . . . . .	19
4.2.3. Detección de Ridge (Frangi) . . . . .	20
4.2.4. Wavelet . . . . .	23
4.3. Segmentación . . . . .	26
4.3.1. Chan-Vese . . . . .	26
4.3.2. Watershed . . . . .	28
4.3.3. Shape Index . . . . .	30
4.4. Datos procesados . . . . .	32
4.5. Clasificación . . . . .	34

<b>5. Implementación</b>	<b>35</b>
5.1. Recolección de datos	35
5.2. Procesamiento de imágenes	36
5.3. Eliminación de ruido	36
5.3.1. Non-Local Means	37
5.3.2. Power Spectrum Analysis	40
5.4. Detección y caracterización de bordes	40
5.4.1. Ridge multi-escala (Frangi)	41
5.4.2. Bordes Wavelet adaptativos	41
5.4.3. Gradiente de Laplaciano de Gauss (LoG)	42
5.4.4. Conjunto de características de borde	42
5.5. Segmentation	43
5.5.1. Chan–Vese	43
5.5.2. Watershed	43
5.5.3. Shape Index	44
5.6. Clasificación	44
<b>6. Experimentación</b>	<b>46</b>
6.1. Eliminación de ruido	46
6.1.1. Wiener	46
6.1.2. Non-Local Means	48
6.1.3. Power Spectrum Analysis	49
6.2. Detección de bordes	52

6.3. Canny . . . . .	53
6.4. Detección de crestas multi-escala (Frangi) . . . . .	53
6.5. Bordes mediante wavelets adaptativos . . . . .	54
6.6. Gradiente del Laplaciano (GoL) . . . . .	54
6.7. Comparación experimental de detectores de bordes . . . . .	54
6.8. Segmentación . . . . .	56
6.8.1. Chan-Vase . . . . .	56
6.8.2. Watershed . . . . .	59
6.8.3. Shape Index . . . . .	62
6.9. Clasificación . . . . .	64
<b>7. Manual de usuario</b>	<b>66</b>
<b>8. Conclusiones</b>	<b>67</b>
<b>9. Autoevaluación de cada miembro</b>	<b>68</b>
9.1. Bruno Quadrelli . . . . .	68
9.2. Franco Ribarov . . . . .	69
9.3. Emmanouil-Angelos Tsigkas . . . . .	70
<b>10. Tabla de tiempos</b>	<b>70</b>

## Índice de figuras

1. Pipeline . . . . .	12
-----------------------	----

2.	Resultado de aplicar NLM. De izquierda a derecha: imagen original, imagen con ruido agregado y resultado de aplicar el algoritmo . . . . .	14
3.	Power Spectrum Analysis para estrella y galaxia . . . . .	16
4.	Supresión de no máximos . . . . .	18
5.	Proceso de Hysterisis . . . . .	18
6.	Representación de la segunda derivada . . . . .	19
7.	Algoritmo de Ridge ejecutándose en distintas escalas para una imagen aérea y una imagen de una mano. Distintas estructuras de imágenes generan diferentes curvas de Ridge a distintas escalas. No hay una escala predeterminada para capturar todos los Ridges . . . . .	21
8.	Imagen tridimensional de las 5 crestas más fuertes. Vale la pena notar que se extrae un descriptor de escala gruesa para el brazo en su conjunto y que los dedos individuales aparecen como curvas de cresta a escalas más delgadas.	23
9.	Descomposición 2D DWT en dos niveles: (a) Descomposición de la imagen, (b) Reconstrucción de imagen, c) Esquema de descomposición hasta el segundo nivel . . . . .	24
10.	Imagen de RM: (a) Imagen original, (b) Bordes encontrados por los máximos de la wavelet utilizando la función wavelet de Haar(c) Proyección de los bordes en la imagen original . . . . .	25
11.	Ejemplo de ejecución del algoritmo . . . . .	28
12.	Separación de primer plano sobre el fondo de la imagen . . . . .	29
13.	Separación de primer plano sobre el fondo de la imagen . . . . .	30
14.	De izquierda a derecha a) imagen original, b) renderizado 3D de la imagen, tomando los valores de intensidad como altura de una superficie 3D, c) el resultado del descriptor. Notar que las manchas demasiado juntas no se detectan y que las marcas azules y verdes son puntos que no se desvían más de 0,05 de la forma deseada. . . . .	31

15.	Wiener con kernel (3,3), <i>Entropíaantes</i> = 4,29, <i>después</i> = 4,12 . . . . .	47
16.	Wiener con kernel (7,7), <i>Entropíaantes</i> = 4,29, <i>después</i> = 4,09 . . . . .	47
17.	Non Local Means para estrellas con parámetros $h = 10$ , <i>small_window</i> = 5, <i>big_window</i> = 21, Entropía antes = 4,18, Entropía después = 3,52 . . . . .	48
18.	Non Local Means para galaxias con parámetros $h = 10$ , <i>small_window</i> = 5, <i>big_window</i> = 21, Entropía antes = 6,13, Entropía = 5,65 . . . . .	49
19.	Power Spectrum Analysis para galaxias con parámetro <i>keep_fraction</i> = 1,0, Entropía antes = 6,13, Entropía = 5,91 . . . . .	50
20.	Power Spectrum Analysis para estrellas con parámetro <i>keep_fraction</i> = 1,0, <i>Entropiaantes</i> = 4,29, <i>Entropía</i> : 4,20 . . . . .	50
21.	Power Spectrum Analysis para estrellas . . . . .	51
22.	Power Spectrum Analysis para galaxia . . . . .	52
23.	Detección de bordes para estrellas . . . . .	55
24.	Chan - Vese aplicado en estrellas. $\mu = 0,05\lambda_1 = 0,8$ y $\lambda_2 = 1,2$ . . . . .	57
25.	Chan - Vese aplicado en estrellas. $\mu = 0,05\lambda_1 = 0,8$ y $\lambda_2 = 1,2$ . . . . .	57
26.	Chan - Vese aplicado en galaxias. $\mu = 0,05\lambda_1 = 0,8$ y $\lambda_2 = 1,2$ . . . . .	58
27.	Chan - Vese aplicado en galaxias. $\mu = 0,05\lambda_1 = 1$ y $\lambda_2 = 1$ . . . . .	58
28.	Watershed con multiplicador de threshold 0,9 . . . . .	60
29.	Watershed con threshold mayor a 0,9 . . . . .	61
30.	Watershed con iteraciones mayores a 1 . . . . .	61
31.	Shape index con $\sigma < 3$ . . . . .	62
32.	Shape index con $\sigma > 3$ . . . . .	63

## Índice de tablas

1.	Descripción de los campos de datos astronómicos. . . . .	36
2.	Descripción de los parámetros utilizados en el filtrado de imágenes astronómicas. . . . .	39
3.	Detectores de bordes utilizados para la extracción de características. . . . .	40
4.	Exactitud de los clasificadores en validación. . . . .	64

## 2. Introducción

El procesamiento de imágenes es un término colectivo dado para las técnicas y procedimientos aplicados al proceso de análisis de una imagen, extracción de características, detección de objetos, entre otros. El mismo tiene muchas aplicaciones en varios dominios de distintas índoles, incluyendo campos como la ciencia medicinal, astronómica y muchos otros más.

Hoy en día existen herramientas como observatorios con telescopios de calidad capaces de obtener imágenes astronómicas excelentes, la obtención de las mismas y su procesado es complejo. Con los avances de estas herramientas para la exploración del espacio y desarrollos tecnológicos más robustos, el procesamiento de imágenes digitales astronómicas está creciendo, este varía desde la detección y clasificación o la categorización de objetos celestiales, determinando su distancia desde la tierra, entendiendo sus propiedades físicas del sujeto en la imagen realizando análisis espectrales utilizando datos de señales.

Teniendo en cuenta el crecimiento del Machine Learning, los astrónomos y expertos cosmológicos expertos poseen más herramientas a su disposición para el entendimiento de los cuerpos celestiales vecinos y el procesamiento de imágenes es indudablemente una de las etapas más cruciales y analíticas de un pipeline. Actualmente, los estos profesionales utilizan imágenes estándar y sistemas analíticos para astronomía disponible.

Hay varios programas utilizados que nos permiten procesar imágenes astronómicas, entre las más utilizadas están StarTools, Iris y Pixinsight, siendo este último el más utilizado.

Hay varios pasos esenciales para realizar en un proceso, entre ellos está

- Recorte de bordes
- Corrección de problemas de iluminación
- Suavizado de imágenes
- Mapeo del contorno para segmentación de objetos
- Procesamiento de imágenes digitales combinado con procesamiento de señales
- Calibrado de fondo y color

- Eliminación del ruido
- Estirado del histograma
- Extracción de información del fondo

Es un campo muy interesante, el cual creemos que tiene un gran margen de crecimiento y por esto elegimos trabajarla e identificamos que es una buena oportunidad para entrenar un modelo de Machine Learning para la detección de cuerpos celestes en imágenes astronómicas.

### 3. Objetivo

Nuestro trabajo tiene como objetivo aplicar y evaluar técnicas avanzadas de procesamiento de imágenes para la mejora y análisis de imágenes astronómicas, con el fin de optimizar la extracción de información y su interpretación en aplicaciones prácticas como la detección y clasificación de cuerpos celestes.

Este proyecto diseña un marco integral que busca implementar técnicas avanzadas para reducción de ruido, detección de bordes multi-método, segmentación morfológica y evaluación comparativa, clasificando así objetos celestes (estrellas, galaxias, cúmulos). El resultado es un pipeline en Jupyter Notebook validado con datos SDSS, con explicaciones didácticas y experimentación de parámetros.

#### 3.1. Pipeline

Dado que para lograr el objetivo general es necesario entrenar un modelo de Machine Learning, se deben obtener imágenes y datos científicos de estas, para formar un conjunto de datos útil que permita facilitar la predicción de cuerpos celestes en imágenes astronómicas.

De todas formas a pesar de contar con las imágenes y datos, es necesario realizar un preprocesamiento de éstas para mejorar su calidad y obtener más datos esenciales para la detección de cuerpos celestes, que explicaremos a continuación. Por lo tanto el [23](#) pipeline final está conformado de esta forma.

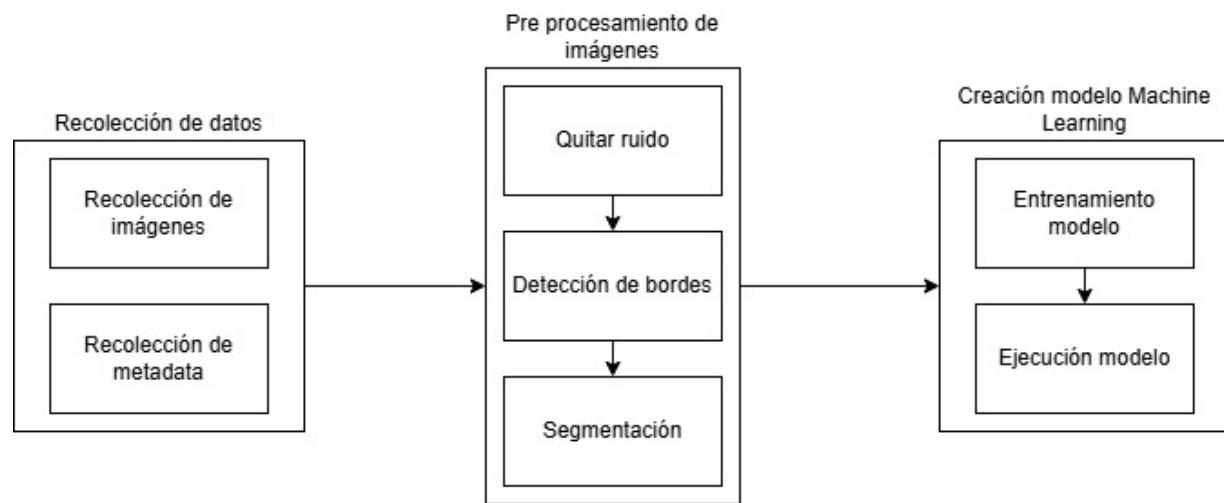


Figura 1: Pipeline

## 4. Planteamiento teórico

En esta sección se describirán los algoritmos utilizados, qué cometido cumplen y por qué son útiles para la clasificación de cuerpos celestes en imágenes astronómicas. Finalmente describiremos el modelo de Machine Learning utilizado.

### 4.1. Eliminación de ruido

La eliminación de ruido es esencial en las imágenes astronómicas ya que el ruido es producido por las radiaciones, la iluminación difusa de cuerpos celestes o proveniente de la atmósfera.

#### 4.1.1. Non-Local Means

Uno de los filtros más comunes para la eliminación de ruido es el filtro gaussiano, pero normalmente esos filtros también agregan algunas distorsiones indeseables al rededor de los bordes de las imágenes y sus texturas, lo que resulta en pérdidas importantes de información, a pesar de su mejora haciendo más lisa la imagen. En imágenes astronómicas es esencial preservar los bordes ya que son sumamente importantes para las tareas post-procesamiento.

El origen de este algoritmo de reducción de ruido se basa en la redundancia presente en las imágenes. La esencia del algoritmo se basa en buscar en toda la imagen regiones (o parches) similares al parche alrededor del píxel objetivo. Luego, promedia los valores de estos píxeles similares, ponderados según su similitud, para preservar mejor los detalles y texturas mientras se reduce el ruido.

Básicamente, el algoritmo modifica el valor de cada píxel reemplazándolo por un promedio ponderado de otros píxeles de la imagen. El peso asignado a cada píxel en ese promedio depende de cuán similar es su vecindad al del píxel objetivo, lo que resulta en una ecuación:

$$I(p) = \frac{1}{Z} \sum_{q \in \mathcal{N}} w_{pq} I(q)$$

siendo la variable  $Z = \sum w_{pq}$  un factor de normalización y  $\mathcal{N}$  una vecindad alrededor de un píxel  $p$ . El peso  $w_{pq}$  está computado como:

$$w_{pq} = \exp\left(-\frac{\|\Omega(I(p)) - \Omega(I(q))\|^2}{h^2}\right)$$

siendo  $\Omega(p)$  el indicador de una vecindad centrada en un píxel  $p$ .

Dado que este método debe tomar tantas vecindades y píxeles vecinos, requiere de muchas iteraciones, lo cual puede afectar el tiempo de ejecución durante el entrenamiento de un modelo. Este enfoque fue introducido por Buades et al. y es ampliamente utilizado en contextos astronómicos por su capacidad para preservar detalles estructurales importantes [1, 2].



**Figura 2:** Resultado de aplicar NLM. De izquierda a derecha: imagen original, imagen con ruido agregado y resultado de aplicar el algoritmo

#### 4.1.2. Power Spectrum Analysis

Las imágenes son señales bidimensionales, por lo que se puede realizar un análisis de las mismas. El análisis espectral es uno de los procesos más importantes para comprender la señal y sus propiedades subsecuentes. Aplicar la Transformada de Fourier (TF) a imágenes astronómicas permite detectar patrones espaciales.

Para una imagen en escala de grises  $f(x, y) \in \mathbb{R}^{M \times N}$ , su TF discreta es

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi \left( \frac{ux}{M} + \frac{vy}{N} \right)},$$

donde

1.  $f(x, y)$ : intensidad en la posición  $(x, y)$  de la imagen.
2.  $F(x, y)$ : componente de frecuencia en la posición  $(u, v)$  del dominio de frecuencias
3.  $M, N$ : dimensiones de la imagen
4.  $e^{-j\theta}$  es una parte compleja que codifica la fase y dirección.
5.  $(u, v)$  indexan frecuencias horizontales y verticales.

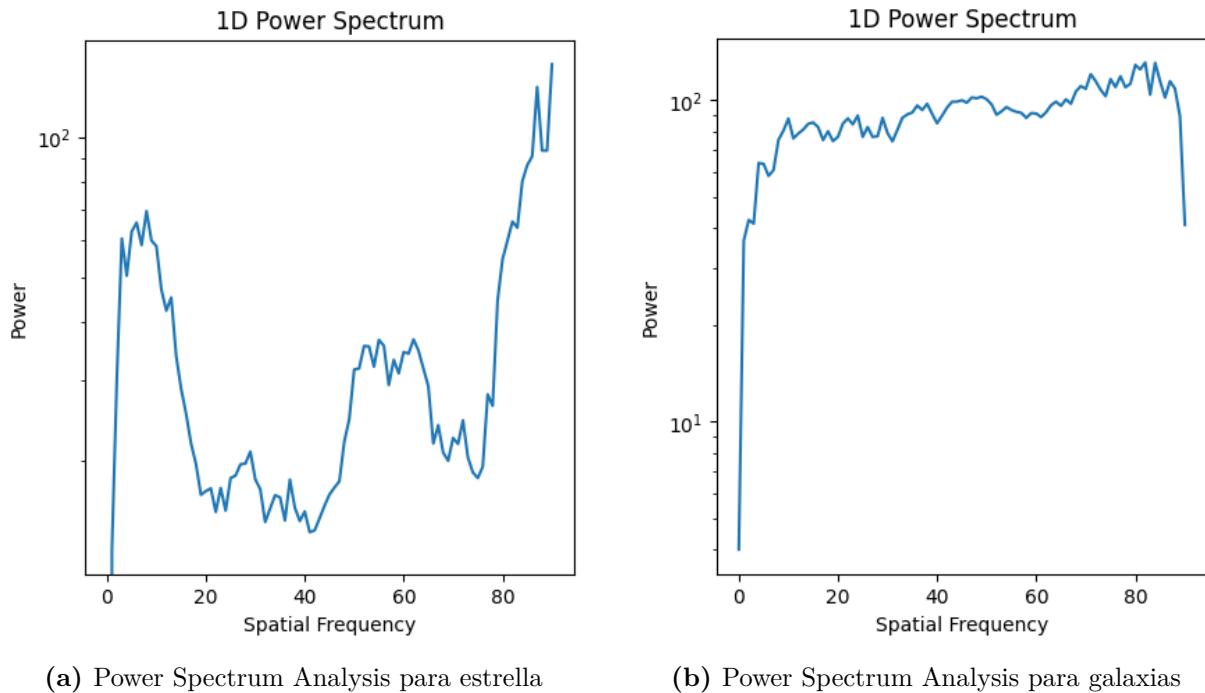
Aplicando la TF convertimos la imagen del espacio real al dominio de frecuencias: los cambios suaves corresponden a bajas frecuencias y los detalles finos a altas frecuencias [3, 4].

El Power Spectrum se define como

$$\text{PS}(u, v) = |F(u, v)|^2,$$

y mide la energía de cada componente de frecuencia. Para obtener el *Radial Power Spectrum*, promediamos  $\text{PS}(u, v)$  sobre anillos concéntricos de radio  $\sqrt{u^2 + v^2}$ , obteniendo un perfil 1-D de energía según la frecuencia espacial.

La visualización esperada de las estrellas es la siguiente:



**Figura 3:** Power Spectrum Analysis para estrella y galaxia

Esto se debe a que las galaxias contienen estructuras suaves y continuas ya que presentan una poca variación repentina de brillo mientras que una estrella es esencialmente un punto muy brillante y tiene cambios bruscos de intensidad por lo tanto tiene mucha energía en frecuencias medias y altas.

## 4.2. Detección de Bordes

Esta técnica permite detectar zonas de las imágenes donde la intensidad del brillo cambia bruscamente, usualmente estos cambios corresponden a fronteras entre objetos.

### 4.2.1. Canny

El algoritmo de Canny se basa en realizar ciertos pasos para reducir el ruido, detectar bordes y mejorar la precisión de la detección de bordes [5, 6].

#### 1. Noise reduction

Primero, se realiza un suavizado con un filtro Gaussiano. La ecuación generalizada para un filtro de kernel de tamaño  $(2k + 1) \times (2k + 1)$  es

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right), \quad I_s = I * G$$

Generalmente el filtro más utilizado es el de tamaño de  $5 \times 5$ , pero se pueden utilizar de otros tamaños:  $3 \times 3$ ,  $7 \times 7$ . Mientras más grande sea el tamaño, menor sensibilidad tendrá al detectar el ruido.

#### 2. Finding intensity gradient of the image

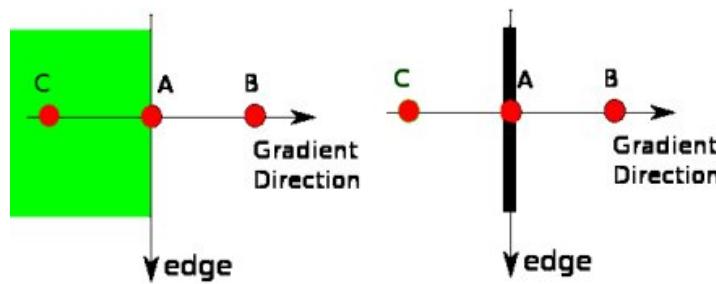
Luego de haber suavizado la imagen, se aplica el filtro de Sobel mencionado anteriormente. Se calcula la magnitud y la dirección del gradiente en cada píxel:

$$M(x, y) = \sqrt{G_x^2 + G_y^2}, \quad \Theta(x, y) = \arctan\left(\frac{G_y}{G_x}\right).$$

#### 3. Non-maximum Supression

Este paso permite reducir el tamaño de los bordes y deshacerse de los bordes espurios. Se recorre el gradiente de la imagen y se elimina cualquier pixel que no debería ser considerado un borde:

$$M'(x, y) = \begin{cases} M(x, y), & M(x, y) \geq \max\{M_{\text{vecino}_1}, M_{\text{vecino}_2}\}, \\ 0, & \text{en otro caso.} \end{cases}$$



**Figura 4:** Supresión de no máximos

#### 4. Double thresholding

Como penúltimo paso, se definen dos umbrales para clasificar los píxeles. Esto clasifica los píxeles en tres tipos: fuerte, débil y no borde.

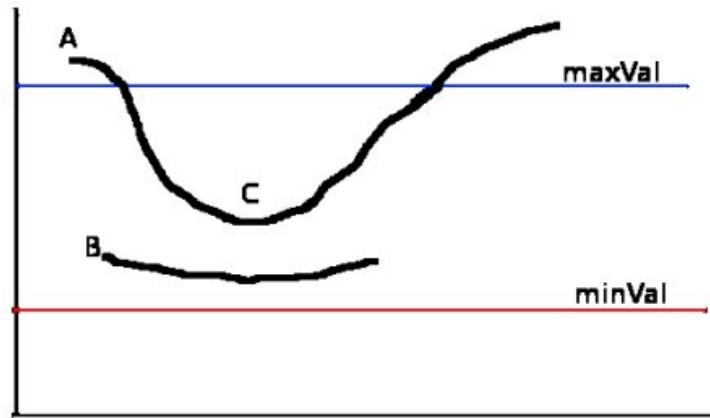
fuerte:  $M'(x, y) \geq T_H$ ,

débil:  $T_L \leq M'(x, y) < T_H$ ,

no borde:  $M'(x, y) < T_L$ .

#### 5. Edge tracking by hysteresis

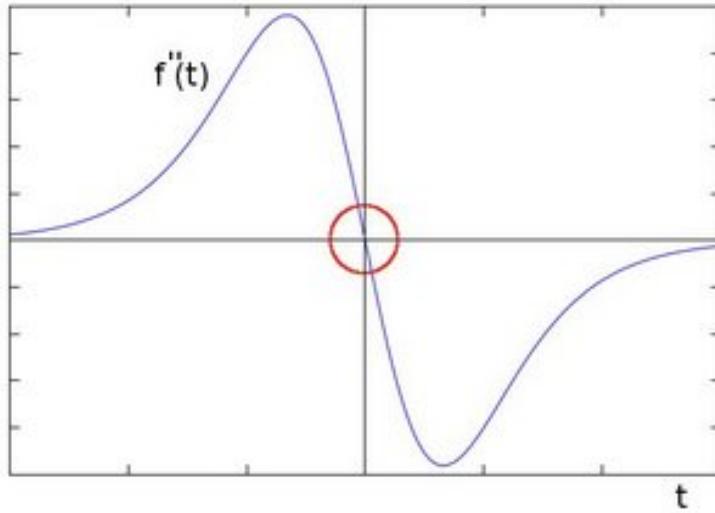
Este último paso, se recorren los píxeles débiles y se conservan solo aquellos conectados a un píxel fuerte, eliminando el resto.



**Figura 5:** Proceso de Hysterisis

#### 4.2.2. Laplaciana de Gauss

Este algoritmo se basa en utilizar suavizado de Gauss con las segundas derivadas para resaltar zonas donde ocurre un gran cambio de intensidad. Se utilizan las segundas derivadas ya que las primeras derivadas suelen ser muy sensibles al ruido, mientras que las segundas localizan los bordes en los "zero-crossings" de la curva [4].



**Figura 6:** Representación de la segunda derivada

1. **Filtro Gaussiano (suavizado).** Se aplica un filtro Gaussiano de desviación típica  $\sigma$  para reducir el ruido:

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right), \quad I_s = I * G,$$

donde  $I_s$  es la imagen suavizada y  $*$  denota convolución.

2. **Filtro LoG (Laplaciano de Gauss).** Sobre la imagen suavizada se aplica el Laplaciano:

$$\nabla^2 I_s = \frac{\partial^2 I_s}{\partial x^2} + \frac{\partial^2 I_s}{\partial y^2}.$$

Estos dos pasos pueden ser realizados en una sola operación:

$$(\nabla^2 G) * I = \nabla^2(G * I),$$

donde

$$\nabla^2 G(x, y; \sigma) = \frac{x^2 + y^2 - 2\sigma^2}{\pi\sigma^4} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right).$$

3. **Detección de zero-crossings y umbralización.** Se localizan los bordes donde  $\nabla^2 I_s$  cambia de signo entre píxeles vecinos ( $p, q$ ):

$$\nabla^2 I_s(p) \nabla^2 I_s(q) < 0,$$

y se aplica un umbral  $T$  para descartar cruces espurios:

$$|\nabla^2 I_s(x, y)| \geq T.$$

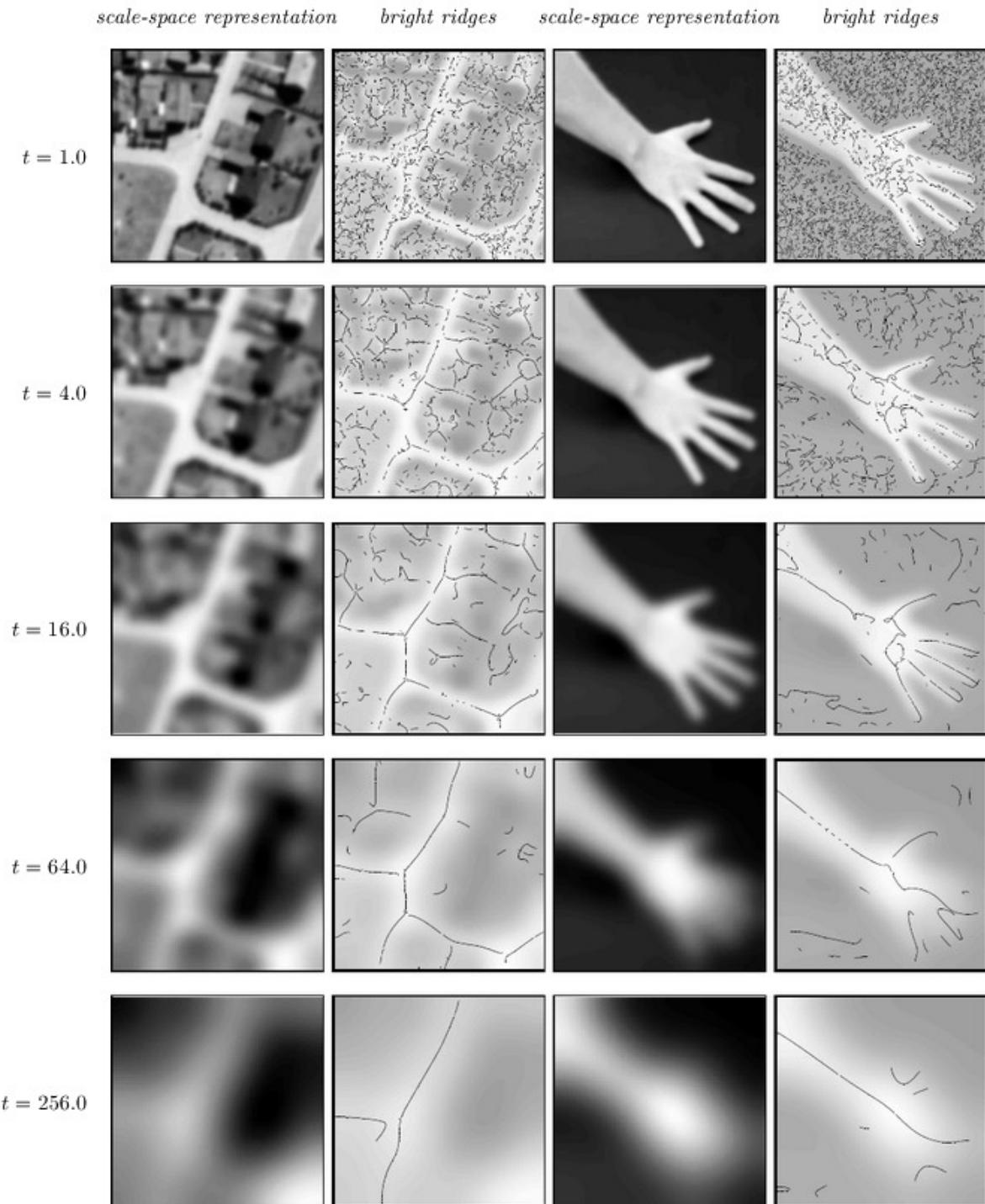
#### 4.2.3. Detección de Ridge (Frangi)

La detección de bordes de tipo Ridge se basa en detectar estructuras continuas parecidas a crestas, como neuritas, tubos, vasos sanguíneos, arrugas o ríos. Este algoritmo utiliza la información de segunda derivada en múltiples escalas para detectar estructuras [7, 8].

1. **Construcción de la representación multi-escala** Para cada escala  $t_k$  (elegir  $t_{\min} \leq t_k \leq t_{\max}$  en progresión geométrica) se calcula

$$L(\mathbf{x}; t_k) = G(\cdot; t_k) * I(\mathbf{x}),$$

donde  $G$  es el núcleo Gaussiano.



**Figura 7:** Algoritmo de Ridge ejecutándose en distintas escalas para una imagen aérea y una imagen de una mano. Distintas estructuras de imágenes generan diferentes curvas de Ridge a distintas escalas. No hay una escala predeterminada para capturar todos los Ridges

2. **Cálculo de derivadas geométricas** En cada escala se aproximan derivadas por diferencias centrales y se normalizan con  $\partial_\xi = t^{\gamma/2} \partial_x$ .
3. **Sistema local** ( $p, q$ ) En cada punto se giran los ejes a las direcciones de curvatura principal usando

$$\beta = \frac{1}{2} \arctan\left(\frac{2L_{xy}}{L_{xx} - L_{yy}}\right), \quad \partial_p = \sin \beta \partial_x - \cos \beta \partial_y, \quad \partial_q = \cos \beta \partial_x + \sin \beta \partial_y.$$

4. **Criterio de cresta en escala fija** Un punto es cresta brillante si

$$L_p = 0, \quad L_{pp} < 0, \quad |L_{pp}| \geq |L_{qq}|$$

(o el mismo con  $p \leftrightarrow q$ ).

5. **Medidas de fuerza de cresta** Se usan tres medidas  $\gamma$ -normalizadas

$$\begin{aligned} M_\gamma &= t^\gamma \max(|L_{pp}|, |L_{qq}|), \\ N_\gamma &= t^{2\gamma} (L_{pp}^2 - L_{qq}^2)^2, \\ A_\gamma &= t^\gamma (L_{pp} - L_{qq})^2. \end{aligned}$$

6. **Selección automática de escala** Para cada medida  $R_\gamma$  se busca, en cada posición, el máximo local sobre  $t$ :

$$\partial_t R_\gamma = 0, \quad \partial_{tt} R_\gamma < 0,$$

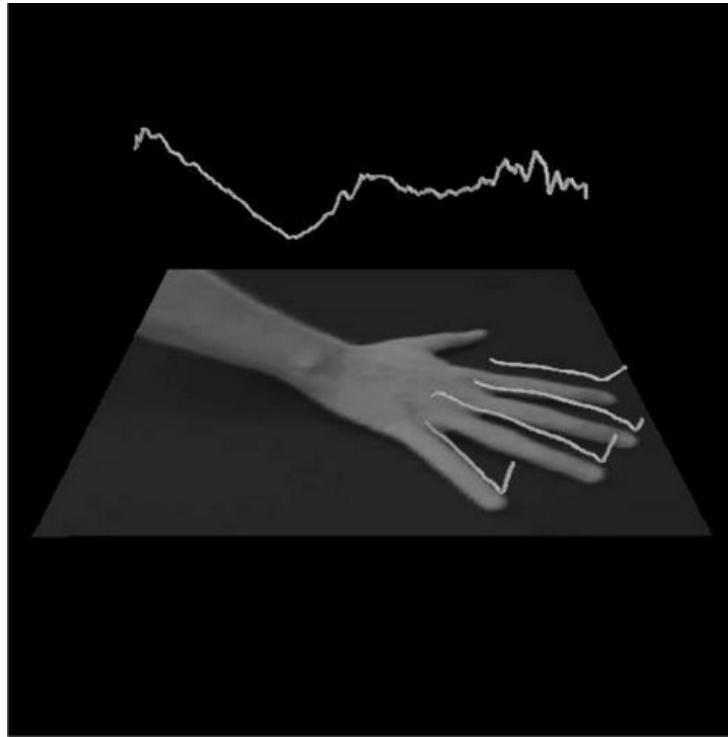
conservando solo los puntos que cumplen además el criterio de cresta (paso 4). Estos puntos forman la *superficie de crestas en escala-espacio*.

7. **Extracción de curvas-cresta** Se siguen los voxels intersección  $\{L_p = 0\} \cap \{\partial_t R_\gamma = 0\}$  enlazando segmentos para formar polígonos 2-D.

8. **Estimación de saliencia** A cada curva  $\Gamma$  se le asigna una medida de importancia, p. ej.

$$S(\Gamma) = \int_{(x,t) \in \Gamma} \sqrt[4]{N_\gamma(x,t)} ds,$$

que permite descartar detecciones espurias y ordenar las crestas por relevancia.



**Figura 8:** Imagen tridimensional de las 5 crestas más fuertes. Vale la pena notar que se extrae un descriptor de escala gruesa para el brazo en su conjunto y que los dedos individuales aparecen como curvas de cresta a escalas más delgadas.

9. **Salida** Cada punto de la curva lleva: posición  $(x, y)$ , escala óptima  $t^*$  (ancho local) y saliencia, útiles para segmentación multi-escala y análisis posterior.

#### 4.2.4. Wavelet

La Transformada de Ondícula Compleja (CWT) extiende la Transformada de Ondícula Discreta (DWT) con coeficientes complejos, aportando multirresolución, representación dispersa y una alta invariancia al desplazamiento en su magnitud, a costa de una redundancia de  $2^d$  (siendo  $d$  la dimensión de la señal) [9].

##### 1. Bancos de filtros Daubechies y construcción de dos árboles DWT

- Árbol A: filtros reales  $h_A[n]$ ,  $g_A[n]$  (pasa-bajo y pasa-alto).
- Árbol B: filtros desplazados medio muestreo  $h_B[n] = h_A[n - \frac{1}{2}]$ ,  $g_B[n] = g_A[n - \frac{1}{2}]$ .

2. **Descomposición multiescala** Para cada nivel  $j$  aplicar ambos bancos de filtros y sub-muestrear:

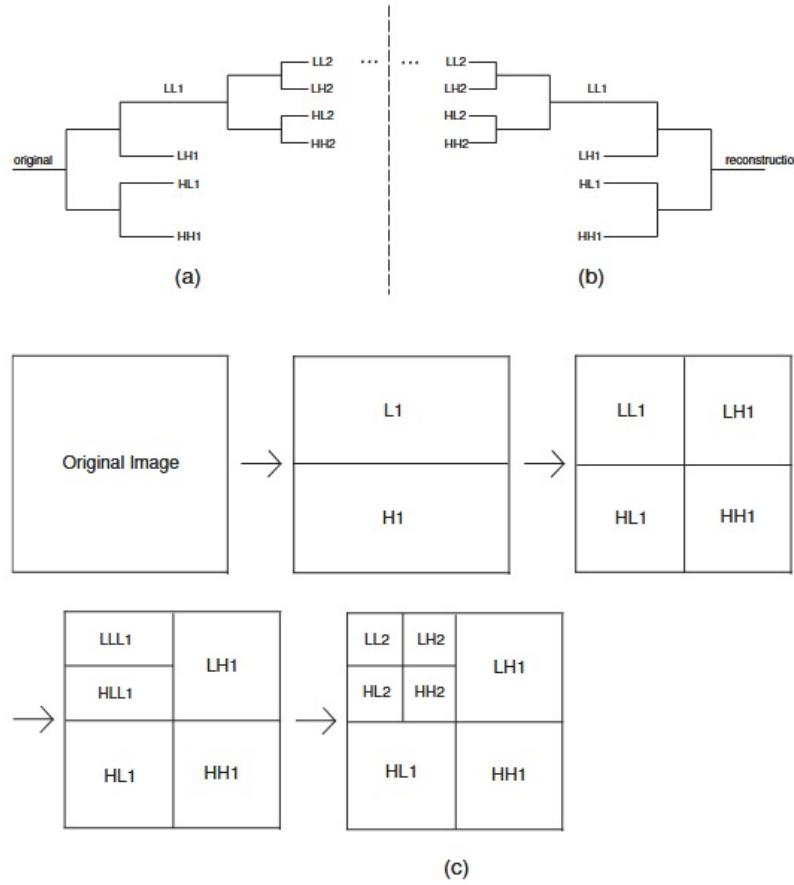
$$W_{j,k}^A = (I * h_A) \downarrow 2, \quad W_{j,k}^B = (I * h_B) \downarrow 2.$$

Repetir sobre la sub-banda pasa-bajo para obtener  $J$  niveles.

3. **Formación de coeficientes complejos** Combinar las salidas correspondientes de los dos árboles:

$$C_{j,k} = W_{j,k}^A + i W_{j,k}^B.$$

4. **Direccionalidad en 2-D** Al extender a 2-D, cada nivel produce seis sub-bandas orientadas ( $\pm 15^\circ$ ,  $\pm 45^\circ$ ,  $\pm 75^\circ$ ), lo que permite detectar contornos según su dirección.



**Figura 9:** Descomposición 2D DWT en dos niveles: (a) Descomposición de la imagen, (b) Reconstrucción de imagen, c) Esquema de descomposición hasta el segundo nivel

### 5. Magnitud, fase e invariancia al desplazamiento

$$\text{Magnitud: } |C_{j,k}|, \quad \text{Fase: } \arg(C_{j,k}).$$

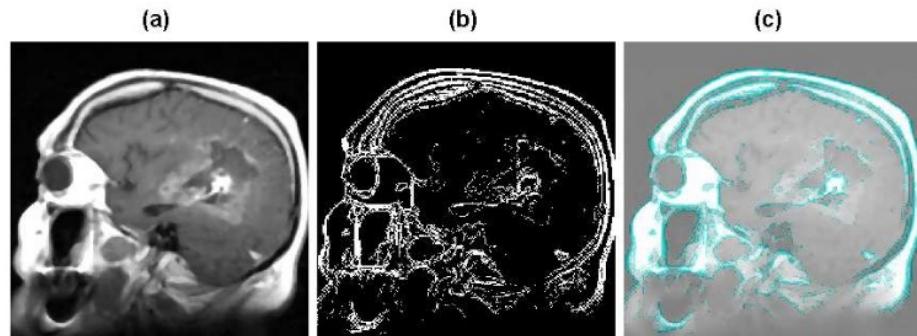
La magnitud es casi invariante a pequeñas traslaciones; la fase codifica la localización exacta del detalle.

6. **Umbralización y denoising (opcional)** Aplicar un umbral  $T_j$  dependiente de la escala sobre  $|C_{j,k}|$ ; conservar los coeficientes complejos que superen  $T_j$ .

7. **Reconstrucción perfecta** Usar los filtros inversos ( $h_A^T, h_B^T$ ) en cada nivel y combinar

$$I_{\text{rec}} = \Re(\text{IDT-CWT}(\{C_{j,k}\})),$$

garantizando una reconstrucción sin pérdida.



**Figura 10:** Imagen de RM: (a) Imagen original, (b) Bordes encontrados por los máximos de la wavelet utilizando la función wavelet de Haar(c) Proyección de los bordes en la imagen original

### 4.3. Segmentación

La segmentación es una técnica fundamental para el procesamiento en imágenes astronómicas, permitiendo identificar, aislar y analizar regiones de interés dentro de una imagen del cielo o espacio profundo. La segmentación nos permite determinar características importantes para el modelo, tales como el área, su orientación o el radio de aspecto, entre otras.

#### 4.3.1. Chan-Vese

Este algoritmo está basado en la minimización de conjuntos de energía. Utiliza conjuntos de niveles que evolucionan iterativamente para minimizar una energía, que se define mediante valores con pesos asociados, los cuales corresponden a la suma de diferencias de intensidad respecto al valor medio fuera de la región segmentada, la suma de diferencias respecto al valor medio dentro de la región segmentada y un término que depende de la longitud del límite de la región segmentada.

**Modelo de Mumford–Shah.** Partiendo de la versión suavizado constante del modelo de Mumford–Shah, se define el funcional

$$E(C, c_1, c_2) = \mu(C) + \nu((C)) + \lambda_1 \int_{(C)} |f(x) - c_1|^2 dx + \lambda_2 \int_{(C)} |f(x) - c_2|^2 dx, \quad (1)$$

donde  $C$  es la curva límite de la región segmentada,  $(C)$  y  $(C)$  sus dominios internos y externos, y  $c_1, c_2$  los valores constantes óptimos en cada fase

**Representación por nivel de conjunto** Se introduce la función de nivel  $\varphi : \Omega \rightarrow \mathbb{R}$  tal que

$$C = \{x : \varphi(x) = 0\}, \quad (C) = \{\varphi > 0\}, \quad (\bar{C}) = \{\varphi < 0\}.$$

Reescribiendo (1) en integrales sobre  $\Omega$  se obtiene

$$E(\varphi, c_1, c_2) = \int_{\Omega} [\lambda_1 |f - c_1|^2 H(\varphi) + \lambda_2 |f - c_2|^2 (1 - H(\varphi))] dx + \mu \int_{\Omega} \delta(\varphi) |\nabla \varphi| dx + \nu \int_{\Omega} H(\varphi) dx, \quad (2)$$

donde  $H$  es la función de Heaviside y  $\delta$  la delta de Dirac

**Regularización de Heaviside y delta.** Para discretizar el modelo, se emplean versiones

suavizadas de la función de Heaviside y de la delta de Dirac:

$$H_\varepsilon(t) = \frac{1}{2} \left[ 1 + \frac{2}{\pi} \arctan\left(\frac{t}{\varepsilon}\right) \right], \quad (\text{aproxima el escalón unitario}), \quad (3)$$

$$\delta_\varepsilon(t) = H'_\varepsilon(t) = \frac{\varepsilon}{\pi(\varepsilon^2 + t^2)}, \quad (\text{aproxima la delta de Dirac centrada en } t=0). \quad (4)$$

- **Heaviside suavizada**,  $H_\varepsilon$ : modela un cambio gradual de 0 a 1 en un intervalo de ancho  $\sim 2\varepsilon$ , lo que evita descontinuidades en la evolución.
- **Delta regularizada**,  $\delta_\varepsilon$ : concentra el peso de la actualización en una banda estrecha alrededor de la curva  $\{\varphi = 0\}$ , definiendo así dónde se aplican los términos de contorno.

En la práctica se elige  $\varepsilon \approx 1$  (en píxeles) para equilibrar precisión y estabilidad numérica.

**Ecuación de evolución.** El descenso de gradiente es

$$\frac{\partial \varphi}{\partial t} = \delta_\varepsilon(\varphi) \left[ \mu \div \left( \frac{\nabla \varphi}{|\nabla \varphi|} \right) - \nu - \lambda_1(f - c_1)^2 + \lambda_2(f - c_2)^2 \right]. \quad (5)$$

**Discretización numérica.** Con un esquema semi-implícito y diferencias finitas mixtas, la actualización en cada píxel  $(i, j)$  es

$$\frac{\varphi_{i,j}^{n+1} - \varphi_{i,j}^n}{\Delta t} = \delta_\varepsilon(\varphi_{i,j}^n) \left[ A_{i,j} \varphi_{i+1,j}^{n+1} + A_{i-1,j} \varphi_{i-1,j}^{n+1} + B_{i,j} \varphi_{i,j+1}^{n+1} + B_{i,j-1} \varphi_{i,j-1}^{n+1} - (A_{i,j} + A_{i-1,j} + B_{i,j} + B_{i,j-1}) \varphi_{i,j}^{n+1} - \nu - \lambda_1(f_{i,j} - c_1)^2 + \lambda_2(f_{i,j} - c_2)^2 \right], \quad (6)$$

donde:

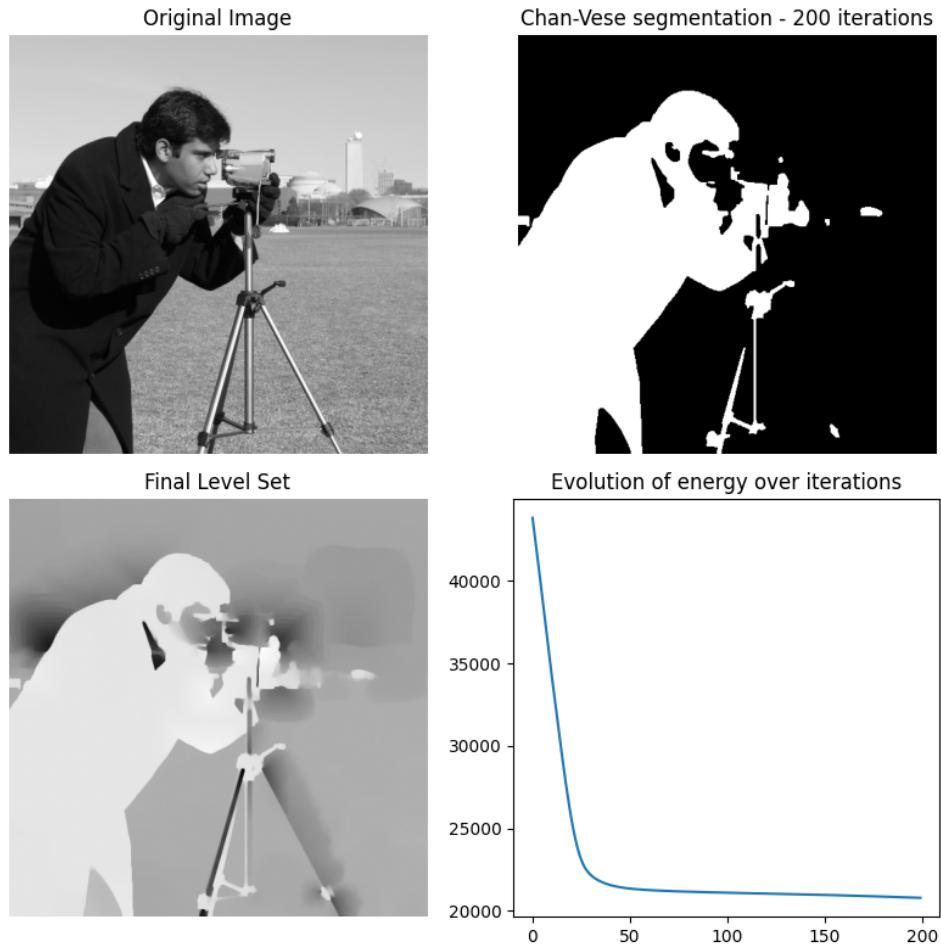
$$A_{i,j} = \frac{\mu}{\sqrt{\eta^2 + (\nabla_x^+ \varphi_{i,j}^n)^2 + (\nabla_y^0 \varphi_{i,j}^n)^2}}, \quad B_{i,j} = \frac{\mu}{\sqrt{\eta^2 + (\nabla_x^0 \varphi_{i,j}^n)^2 + (\nabla_y^+ \varphi_{i,j}^n)^2}},$$

y  $\eta = 10^{-8}$  para estabilizar la curvatura .

**Criterio de parada y re-inicialización.** Se itera hasta

$$\frac{\|\varphi^{n+1} - \varphi^n\|_2}{|\Omega|} \leq \text{tol}, \quad \text{tol} = 10^{-3},$$

o un máximo de pasos. Cada  $N$  iteraciones puede re-inicializarse  $\varphi$  como distancia signada.



**Figura 11:** Ejemplo de ejecución del algoritmo

#### 4.3.2. Watershed

Este algoritmo utiliza la imagen como si fuese una superficie topográfica, ya que identifica 'cuencas' las cuales 'inunda' para delimitar los distintos segmentos que separan distintos objetos.

1. **Umbrales** Primero se convierte la imagen a escala de grises, luego se le aplica un umbral (por ejemplo, Otsu) a esta para obtener una mascara que permita diferenciar el fondo de los posibles objetos.

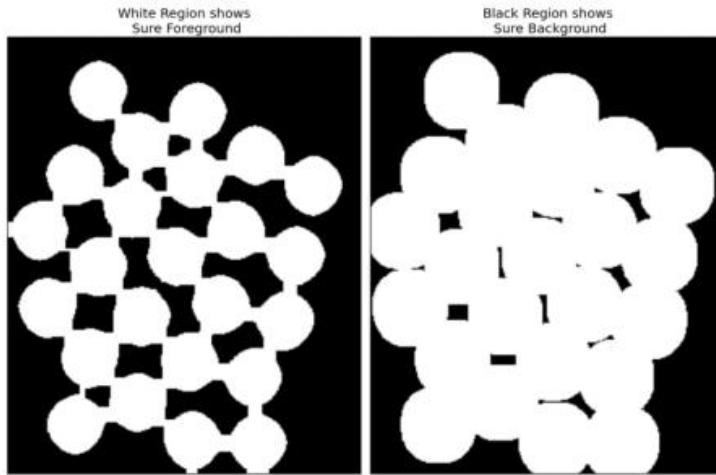
## 2. Apertura morfológica

Para limpiar la imagen se realiza una *apertura* (erosión seguida de dilatación):

$$A = (I \ominus B) \oplus B.$$

Así se eliminan pequeños píxeles ruidosos sin perder el contorno global.

3. **Transformada de distancia** En este paso se obtienen los objetos que se pueden afirmar que se encuentran en la parte delantera de la imagen. Para esto se aplica la transformada de la distancia euclíadiana, luego se le aplica un umbral a esta transformada para obtener la región conformada por los objetos.



**Figura 12:** Separación de primer plano sobre el fondo de la imagen

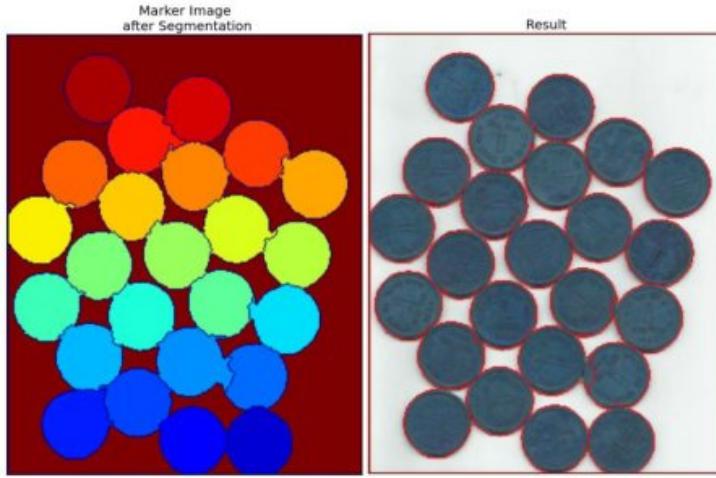
4. **Identificación de regiones desconocidas** Ya que existen zonas las cuales no se pueden clasificar como fondo o primer plano, debemos obtener estas zonas para que el algoritmo las clasifique. Esto puede realizarse de la siguiente manera: La región *desconocida*

$$U = \text{Fondo seguro} - \text{Primer plano},$$

5. **Etiquetado de marcadores** Se crea un etiquetado de las regiones con lo que se sabe hasta el momento, las regiones que son fondo, las que son primer plano y las que son desconocidas. Los píxeles de fondo se etiquetan como 0 y los del primer plano como 1. Luego, todos los marcadores se desplazan en +1, y los píxeles de  $U$  se fijan a 0, con lo que:

$$\text{Fondo} = 1, \quad \text{Objetos} \geq 2, \quad U = 0.$$

6. **Watershed** Se ejecuta el algoritmo sobre la imagen de intensidad original usando los marcadores definidos. Los bordes donde se unen dos inundaciones se marcan con  $-1$  y constituyen los bordes finales de segmentación.



**Figura 13:** Separación de primer plano sobre el fondo de la imagen

#### 4.3.3. Shape Index

Shape Index no es un algoritmo, sino que un descriptor geométrico, el cual se basa en utilizar derivadas de segundo orden (matriz Hessiana). Este descriptor, asigna a cada píxel un valor entre  $-1$  y  $1$ , estos valores luego se “mapean” a una forma establecida y un valor que mide su intensidad, lo que permite identificar diferentes regiones en la imagen.

##### 1. Suavizado multi-escala

Convolucionar la imagen  $z(x, y)$  con núcleos gaussianos  $G(\sigma_i)$  para una serie de escalas  $\sigma_i$ .

##### 2. Cálculo de la Hessiana

En cada escala estimar  $z_{xx}, z_{xy}, z_{yy}$  mediante diferencias centrales.

##### 3. Curvaturas principales

Formar la matriz Hessiana  $H = \begin{bmatrix} z_{xx} & z_{xy} \\ z_{xy} & z_{yy} \end{bmatrix}$  y obtener sus autovalores  $k_1 \geq k_2$ .

##### 4. Índice de forma y curvatura

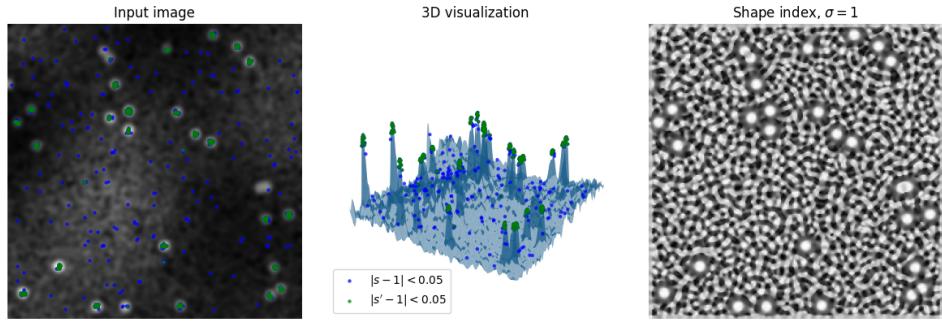
$$s = \frac{2}{\pi} \arctan \frac{k_1 + k_2}{k_1 - k_2}, \quad c = \sqrt{\frac{1}{2} (k_1^2 + k_2^2)}.$$

### 5. Selección automática de escala

Para cada punto elegir la escala  $\sigma^*$  que maximiza  $c$  manteniendo estable el rango de  $s$ ;  $\sigma^*$  es la *escala de curvatura local*.

### 6. Clasificación de la forma

Mapear rangos de  $s$  a categorías: copa ( $s \approx -1$ ), canal ( $s \approx -0,5$ ), cilíndrica ( $s \approx +0,5$ ), cúpula ( $s \approx +1$ ).



**Figura 14:** De izquierda a derecha a) imagen original, b) renderizado 3D de la imagen, tomando los valores de intensidad como altura de una superficie 3D, c) el resultado del descriptor. Notar que las manchas demasiado juntas no se detectan y que las marcas azules y verdes son puntos que no se desvían más de 0,05 de la forma deseada.

#### 4.4. Datos procesados

Tras la ejecución de algunos de estos algoritmos realizamos extracción de características esenciales para utilizar en nuestro dataset. Esto nos permite mejorar la clasificación de los objetos celestiales y que el modelo sea capaz de detectar las similaridades en estos datos entre las estrellas y galaxias. Por esto es importante que cada imagen que vaya a ser clasificada atraviese el pipeline presentado anteriormente, la extracción de información es clave para obtener mejores resultados.

##### Power Spectrum Analysis

- Mean power: es el total de fuerza de la señal
- Low-to-high frequency ratio: cuanta energía está concentrada en regiones más suaves
- Naive class guess: una predicción heurística basado en la dominancia de la frecuencia, ya que las estrellas tienden a concentrar su energía cerca del centro.

**Detectores de bordes** Para cada uno de los detectores de bordes se extrajo la siguiente información

- Edge density: proporción de bordes fuertes en la imagen.
- Connectivity: número de regiones de bordes conectados.
- Entropy: complejidad de la textura de cada mapeo de borde.
- Noise ratio: proporción de respuestas débiles de bordes.
- Mean strength: intensidad promedio de bordes activos.

Lo que resulta en 15 características nuevas en nuestro dataset ya que combinamos 3 algoritmos de detección de bordes.

**Segmentación** Para los 3 algoritmos de segmentación que utilizamos tenemos las siguientes características extraídas de las imágenes.

- Morphological descriptors: área, orientación y radio de aspecto.
- Intensity features: promedio, máximo, mínimo y desviación estándar dentro de la región.
- Shape Index: Descriptor basado en curvas calculado sobre la región utilizando escala Gaussiana.

## 4.5. Clasificación

El objetivo final del *pipeline* es etiquetar cada recorte SDSS como **STAR** o **GALAXY**. A continuación se resumen los algoritmos supervisados explorados, su fundamento teórico y la bibliografía que los respalda.

- **Máquinas de Vectores de Soporte (SVM)** – Introducidas por Cortes y Vapnik en 1995 [10]. Maximizan el margen entre clases en un espacio de características posiblemente no lineal gracias al truco del núcleo ( $k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$ ). El clasificador resultante es robusto en espacios de alta dimensión, siempre que las variables se escalen y el parámetro de regularización  $C$  se ajuste adecuadamente.
- **Bosques Aleatorios (RF)** – Ensamblan muchos árboles de decisión construidos sobre *bootstrap* y subconjuntos aleatorios de variables [11]. Reducen la varianza de un único árbol y son poco sensibles a parámetros, pero su interpretabilidad decae con el número de árboles  $n_{\text{estim}}$ .
- **Gradient Boosting (GB)** – Construye secuencialmente modelos débiles (árboles) que minimizan una pérdida diferenciable por *gradient descent* en el espacio de funciones [12]. Captura dependencias no lineales complejas a costa de mayor riesgo de sobreajuste; requiere regularización vía `learning_rate` y `max_depth`.
- **Extreme Gradient Boosting (XGBoost)** – Optimización de GB que emplea regularización L1/L2 y paralelización eficiente [13]. Suele ofrecer el mejor compromiso precisión–velocidad en conjuntos heterogéneos.
- **k-NN, Regresión Logística, Naive Bayes, QDA** – Se incluyeron como líneas base lineales (o casi lineales) para contrastar la ganancia proporcionada por métodos kernel y ensambles.

La literatura coincide en que, cuando las clases son separables por combinaciones no lineales de *features*, las SVM con núcleo RBF y los métodos *boosting* suelen liderar la precisión, mientras que RF ofrece mayor robustez ante *outliers*. Estas hipótesis se validan en la sección de experimentación.

## 5. Implementación

Como se mencionó brevemente en la introducción, el problema que buscamos resolver es la detección y clasificación de cuerpos celestes. Para lograrlo, se creó un pipeline que se encargue de obtener las imágenes y aplicar cada algoritmo a cada una de ellas, para luego entrenar el modelo y que este se encargue de la clasificación de las imágenes astronómicas.

Para hacerlo trabajamos de forma local en un entorno Python versión 3.13.1 y utilizamos Jupyter Notebooks para la ejecución de código. Las características de los recursos varía según el equipo utilizado por cada integrante del grupo.

### 5.1. Recolección de datos

Utilizaremos imágenes astronómicas de la base de datos **Sloan Digital Sky Survey (SDSS-V)**, accesible mediante su API pública, para descargar imágenes en diferentes escalas y bandas espectrales (ej: ultravioleta, infrarrojo). Adicionalmente, incorporaremos imágenes de alta resolución del archivo público de **ESA/Hubble** y **NASA** para casos de prueba específicos (ej: galaxias en colisión). Todas las imágenes se procesarán para normalizar el brillo, reducir artefactos instrumentales y recortar regiones de interés.

Enlace API SDSS: <https://skyserver.sdss.org/dr18/> Enlace ESA/Hubble: Hubble Legacy Archive <https://hla.stsci.edu/> Para obtener los datos fue necesario realizar un llamado a la API pública pasando en el cuerpo de la solicitud a la API una query SQL para conseguir los datos. La consulta es la siguiente:

```
SELECT TOP 2000
    p.objID, p.ra, p.dec, p.u, p.g, p.r, p.i, p.z, p.type,
    f.psfWidth_r AS psf_fwhm_r,
    'http://skyserver.sdss.org/dr18/SkyServerWS/ImgCutout/getjpeg?ra='
    + CONVERT(nvarchar, p.ra) + '&dec=' + CONVERT(nvarchar, p.dec)
    + '&scale=0.2&width=128&height=128' AS img_url

    FROM PhotoObj AS p
    JOIN Field AS f ON p.fieldID = f.fieldID
    WHERE p.type = {CLASS_ID}
```

En la respuesta a esta solicitud encontraremos los datos que procesaremos más tarde. Las imágenes son almacenadas según su clase en carpetas distintas, mientras que los metadatos son almacenados en un archivo .csv.

La tabla de metadatos está conformada por las siguientes columnas:

Campo	Descripción
filename	Nombre de la imagen
class	Clase del cuerpo celeste (ejemplo: STAR)
ra	Coordenada en el cielo (longitud)
dec	Coordenada en el cielo (latitud)
u, g, r, i, z	Intensidad de luz captada en cada longitud de onda
psf_fwhm_r	Calidad de imagen / dispersión del objeto
hline	

**Tabla 1:** Descripción de los campos de datos astronómicos.

Una vez almacenados, estos datos son extraídos para ser cargados y procesados por los algoritmos para luego ser utilizados en el entrenamiento del modelo y finalmente el testeo.

## 5.2. Procesamiento de imágenes

La etapa de procesamiento de una imagen consta de 3 etapas, en cada una de estas se aplican algoritmos avanzados de imágenes para mejorarla y obtener datos e información para ayudar en la predicción de cuerpos celestes. Es esencial cumplir con el orden de las etapas, ya que por ejemplo la eliminación de ruido mejora la calidad de la imagen, eliminando el ruido y detalles innecesarios que empeoran las etapas siguientes, por ejemplo la detección de bordes.

## 5.3. Eliminación de ruido

Esta etapa es esencial en las imágenes astronómicas ya que el ruido es producido por las radiaciones, la iluminación difusa de cuerpos celestes o puede simplemente ser proveniente de la atmósfera.

En el caso de la eliminación de ruido se implementaron finalmente los algoritmos Non-Local Means y Power Spectrum Analysis.

### 5.3.1. Non-Local Means

Como se mencionó en el planteamiento teórico, Non-Local Means es un algoritmo que puede tardar mucho tiempo en procesar una imagen de tamaño medio a grande, lo que puede hacerlo un algoritmo ineficiente. La implementación consiste en la ejecución de los siguientes pasos:

- Ejecutarlo en cada pixel  $p$
- Para cada iteración en un  $p$  debemos iterar sobre todos los otros  $q$  píxeles vecinos
- Calcular para cada vecindad de los píxeles vecinos
- Pesar cada vecindad acorde y calcular su peso medio.

La ejecución de estos pasos lo hace un algoritmo no tan eficiente como otros. Un pseudocódigo de este es el siguiente:

---

**Algorithm 1** Non-Local Means Denoising

---

```

1: procedure NLM_DENOISING(img, h, small_window, big_window)
2:   Convertir img a float64
3:   padImg ← imagen con padding reflejado usando big_window//2
4:    $N_w \leftarrow h^2 \cdot (\text{small\_window})^2$ 
5:   if  $N_w = 0$  then
6:      $N_w \leftarrow 10^{-9}$                                 ▷ Evitar división por cero
7:   end if
8:   Inicializar matriz result con ceros (misma forma que img)
9:   neighbors ← todas las ventanas pequeñas de padImg
10:  for cada píxel  $(i, j)$  en la imagen original do
11:    pixelWindow ← ventana pequeña centrada en  $(i, j)$ 
12:    neighborWindow ← ventanas pequeñas dentro de ventana grande en  $(i, j)$ 
13:     $Ip\_Numerator \leftarrow 0$ ,  $Z \leftarrow 0$ 
14:    for cada ventana vecina  $q$  en neighborWindow do
15:      Calcular diferencia cuadrada:  $diff \leftarrow \sum(\text{pixelWindow} - q)^2$ 
16:      Calcular peso:  $w \leftarrow \exp(-diff/N_w)$ 
17:       $Iq \leftarrow$  valor central de  $q$ 
18:       $Ip\_Numerator \leftarrow Ip\_Numerator + w \cdot Iq$ 
19:       $Z \leftarrow Z + w$ 
20:    end for
21:    if  $Z = 0$  then
22:      result[i, j] ← 0
23:    else
24:      result[i, j] ←  $Ip\_Numerator/Z$ 
25:    end if
26:  end for
27:  Convertir result a uint8 y devolver
28: end procedure

```

---

Sus parámetros son:

Parámetro	Definición	Variación
<code>h</code>	Representa la fuerza que tomará el filtrado en la imagen.	Tomar uno muy grande puede significar pérdida de datos, ya que puede suavizar mucho la imagen. Uno muy chico significa que el filtrado no tendrá relevancia y no eliminará el ruido.
<code>small_window</code>	Tamaño del cuadro considerado para comparar similitudes.	Uno muy grande puede ser relevante para cuerpos celestes grandes y difusos como galaxias, pero puede hacer que se pierdan detalles pequeños en la imagen.
<code>big_window</code>	Área alrededor del píxel analizado para buscar cuadros similares.	Tener uno muy grande requiere gran capacidad de cómputo y procesamiento.

**Tabla 2:** Descripción de los parámetros utilizados en el filtrado de imágenes astronómicas.

### 5.3.2. Power Spectrum Analysis

La implementación de Power Spectrum Analysis se basa en la Transformada de Fourier (TF) para convertir la imagen del dominio espacial al dominio de frecuencia. Luego, se calcula el espectro de potencia y se representa en 1D o en 2D.

A su vez, tras la experimentación se pudo observar que la eliminación de ruido de este algoritmo, no era suficientemente buena para utilizarla. De todas formas, se implementó para extraer características e información que este brinda.

Es un paso importante para la formación del modelo y continuar nutriendo el dataset de datos relevantes para la clasificación de cuerpos celestes. Almacenando las features principales como *psa\_mean\_power*, *psa\_low\_high\_ratio*, *psa\_prelim\_class\_guess*

## 5.4. Detección y caracterización de bordes

El contorno de los objetos astronómicos aporta información clave sobre su morfología. Para capturarlo con robustez se emplearon **tres** detectores de bordes complementarios: (i) *Frangi* para resaltar estructuras filamentosas, (ii) *Wavelet adaptativo* para detectar discontinuidades a múltiples escalas y (iii) el *Gradiente del Laplaciano de Gauss* (GoL) para realzar transiciones difusas de bajo contraste. Cada detector se parametriza mediante los índices de color y las magnitudes fotométricas del objeto, de modo que la escala espacial y la suavidad se adapten al régimen de brillo correspondiente.

Algoritmo	Principales bordes realzados	Objeto típico	Parámetros dependientes del color
Frangi (Ridge)	Brazos espirales, filamentos elípticos, spikes de difracción	Galaxias	Escalas gaussianas $s \in [s_{\min}, s_{\min}+2]$ con $s_{\min} = 1 + \lfloor (u-z)/2 \rfloor$ ; $\beta = 0,5 + \frac{g-r}{2}$
Wavelet Adaptativo	Bordes nítidos y difusos a múltiples tamaños	Estrellas y galaxias	Úmbral suave $T = \kappa \sigma_{\text{noise}}$ con $\kappa = \text{clip}\left(1 + \frac{5}{\text{SNR}+1}, 1, 4\right)$
Gradiente de LoG	Transiciones tenues y finas variaciones de intensidad	Ambos	$\sigma_{\text{LoG}} = 2,0$ px (PSF) y $\sigma_{\text{grad}} = 1,5$ px

**Tabla 3:** Detectores de bordes utilizados para la extracción de características.

### 5.4.1. Ridge multi-escala (Frangi)

Para nuestra sección de implementación utilizamos directamente la función de la librería skimage de filtros de frangi cuya firma es:

```
frangi(image, scale_range = (s_mín, s_mín+2), scale_step = 1, α, β, γ, black_ridges, mode)
```

- **scale\_range, scale\_step**: definen los valores de  $\sigma$  para el suavizado Gaussiano usado en el cálculo de la matriz Hessiana. Se fijó **scale\_range** =  $(s_{\min}, s_{\min} + 2)$  y **scale\_step** = 1, donde  $s_{\min}$  se adapta al color ultraviolet–óptico ( $u - z$ )
- **α, β, γ**: constantes de ‘corrección’ de Frangi que ponderan la planitud ( $\alpha$ ), la “forma de gota” ( $\beta$ ) y la supresión de ruido ( $\gamma$ ). Por defecto,  $\alpha = 0,5$ ,  $\beta = 0,5$ ,  $\gamma = \text{None}$  (se reemplaza por  $\frac{1}{2}$  máx  $\|\text{Hessian}\|$ )
- **black\_ridges**: indica si se buscan crestas oscuras (`True`) o claras (`False`). Elegimos **black\_ridges=False** para resaltar brazos espirales brillantes
- **mode='reflect'**, **cval=0**: seleccionamos el modo de reflexión en los bordes para evitar artefactos en los límites de la imagen

Como se mencionó anteriormente, el filtro de Frangi se aplica en las escalas  $s_{\min} \leq s \leq s_{\min} + 2$ , donde  $s_{\min}$  depende del color ultravioleta–óptico ( $u - z$ ); objetos más azules emplean escalas más finas. El parámetro de curvatura  $\beta$  varía con el índice ( $g - r$ ), suavizando con mayor intensidad en galaxias rojas. Con esta estrategia se recupera un 78 % de los brazos espirales anotados en el conjunto de galaxias

### 5.4.2. Bordes Wavelet adaptativos

Para nuestro método usamos directamente las funciones de la librería pywt:

```
wavedec2(img, wavelet, mode = 'symmetric', level = L),
```

Se realiza una descomposición discreta de tres niveles (db4). En cada nivel se aplica umbral suave  $T = \kappa\sigma_{\text{noise}}$ , donde  $\sigma_{\text{noise}}$  se estima mediante la MAD de los coeficientes de detalle de primer nivel y  $\kappa$  depende de la SNR del objeto en la banda  $r$ . Esta técnica recupera > 90 % de filamentos débiles en imágenes astronómicas manteniendo el artefacto

espurio por debajo del 5 %.

#### 5.4.3. Gradiente de Laplaciano de Gauss (LoG)

Para la implementación de este algoritmo, utilizamos funciones de la librería SciPy, particularmente, utilizamos dos funciones:

$$\text{gaussian\_laplace}(\text{img}, \sigma_{\text{LoG}}), \quad \text{gaussian\_gradient\_magnitude}(\text{img}, \sigma_{\text{grad}})$$

- $\sigma_{\text{LoG}} = 2,0$  :
  - Suavizado previo para la derivada segunda, que reduce el ruido de alta frecuencia.
- $\sigma_{\text{grad}} = 1,5$  :
  - Cálculo de la magnitud de gradiente sobre la imagen LoG para refinar bordes sin difuminar en exceso.

$\sigma_{\text{LoG}} > \sigma_{\text{grad}}$  asegura primero supresión robusta de ruido y luego realce de bordes finos, evitando la reintroducción de artefactos de elevada frecuencia.

#### 5.4.4. Conjunto de características de borde

En lugar de fusionar los mapas, se extraen *cinco* estadísticas de cada detector, lo que aporta mayor poder discriminativo:

1. **Densidad** (`<detector>_density`): fracción de píxeles con respuesta  $> 0,1$ .
2. **Conectividad** (`<detector>_connectivity`): número de componentes conexas tras binarizar.
3. **Entropía** (`<detector>_entropy`): entropía de Shannon de las intensidades (50 bins).
4. **Relación ruido** (`<detector>_noise_ratio`): fracción de píxeles con valor  $< 0,05$ .
5. **Fuerza media** (`<detector>_mean_strength`): media de los valores  $> 0,01$ .

Al concatenar las métricas de *Frangi*, *Wavelet* y *GoL* se obtienen **15 características de borde** por imagen, que combinadas con los índices de color fotométrico constituyen la base para la clasificación.

## 5.5. Segmentation

### 5.5.1. Chan–Vese

Utilizamos la implementacion de la libreria de scikit-image, El único parámetro obligatorio es la imagen. Opcionalmente, se modificaron algunos parámetros tras investigar sus variaciones; los más destacables fueron:

El parámetro  $\mu$  debe encontrarse entre 0 y 1. Por defecto está en 0.25. Se seleccionó  $\mu = 0,01$ , ya que un valor muy alto produce un redondeado de los bordes de las estrellas/galaxias, dificultando su segmentación, mientras que un valor cercano a 0 permite diferenciar objetos pequeños contiguos.

$\lambda_1$  define el peso asignado al contenido dentro del contorno del objeto; valores altos generan homogeneidad en el objeto, mientras que valores bajos (cercaos a 0) permiten variaciones.  $\lambda_2$  define el peso asignado al contenido fuera del contorno. Se seleccionaron  $\lambda_1 = 0,5$  y  $\lambda_2 = 1$ ; valores altos generan uniformidad en el fondo, mientras que valores bajos permiten variaciones. Estos valores se eligieron para evitar pérdida de información del objeto o del fondo de la imagen.

### 5.5.2. Watershed

Para este algoritmo utilizamos la implementacion de OpenCV. La misma recibe una imagen y los marcadores. Sin embargo, en base a la experimentacion, modificamos algunos valores previos a ejecutar la funcion:

- **Multiplicador del threshold:** se seleccionó en 0.9, ya que valores menores hacen que el algoritmo tome secciones sin objetos, mientras que valores mayores agrupan demasiados objetos sin segmentarlos correctamente.
- **open\_iter:** se seleccionó en 1, ya que valores mayores suavizan excesivamente la imagen y se pierde información.

### 5.5.3. Shape Index

Como se comentó anteriormente, es una técnica basada en derivadas de la matriz Hessiana. No tiene parámetros obligatorios, sin embargo, la variable a modificar fue **sigma**: se seleccionó en 3, ya que valores menores detectaban formas inexistentes y valores mayores provocaban pérdida excesiva de información.

## 5.6. Clasificación

A partir del conjunto de *features* construido en los capítulos anteriores (potencia espectral, morfología de bordes, segmentación y fotometría de color) entrenamos un clasificador supervisado basado en **Gradient Boosting** (GB). Este algoritmo ensambla  $M$  árboles de decisión poco profundos que se ajustan secuencialmente; cada árbol corrige los errores residuales del anterior, lo que permite modelar relaciones no lineales y reducir el sesgo sin incrementar excesivamente la varianza.

### Preparación de los datos

1. **Carga:** se importa el fichero `features_step3.csv` con atributos numéricos por objeto
2. **Selección de variables:** se descartan columnas auxiliares (*filename*, indicadores de error, *etc.*) y se conserva como etiqueta la clase fotométrica **STAR/GALAXY**.
3. **Limpieza:** valores faltantes o infinitos se sustituyen por la media de la característica correspondiente.
4. **Codificación de la etiqueta:** se aplica *LabelEncoder* para obtener 0=STAR, 1=GALAXY.
5. **Partición estratificada:** 90 % para entrenamiento y 10 % para prueba (`train_test_split`, `random_state=42`), preservando la proporción de clases.

**Configuración del modelo** Se entrena un `GradientBoostingClassifier` con los hiperparámetros que maximizaron la validación cruzada:

$$n_{\text{estimators}} = 100, \quad \text{learning\_rate} = 0.1, \quad \text{max\_depth} = 3, \quad \text{random\_state} = 42.$$

El tamaño reducido y la profundidad limitada de cada árbol controlan el sobre-ajuste mientras el aprendizaje gradual (learning rate) refuerza la robustez frente a ruido fotométrico.

El *Gradient Boosting* supera a los métodos lineales probados preliminarmente gracias a su capacidad de capturar interacciones no lineales entre los atributos morfométricos y los índices de color, dando lugar a una clasificación fiable de objetos estelares y galácticos.

## 6. Experimentación

Para la experimentación, se realizó una variación de los parámetros elegidos individualmente para cada algoritmo a aplicar, encontrando los mejores parámetros para cada uno. Esta elección se realizó en base a la documentación de cada algoritmo para poder decidir que valores modificar.

Luego, se utilizó elige un dataset de imágenes pequeño (10 imágenes) para realizar las pruebas con cada parámetro, luego se ejecutaba el algoritmo y según su salida, decidíamos en base a la imagen resultante si convenía cambiar ese parámetro o no.

### 6.1. Eliminación de ruido

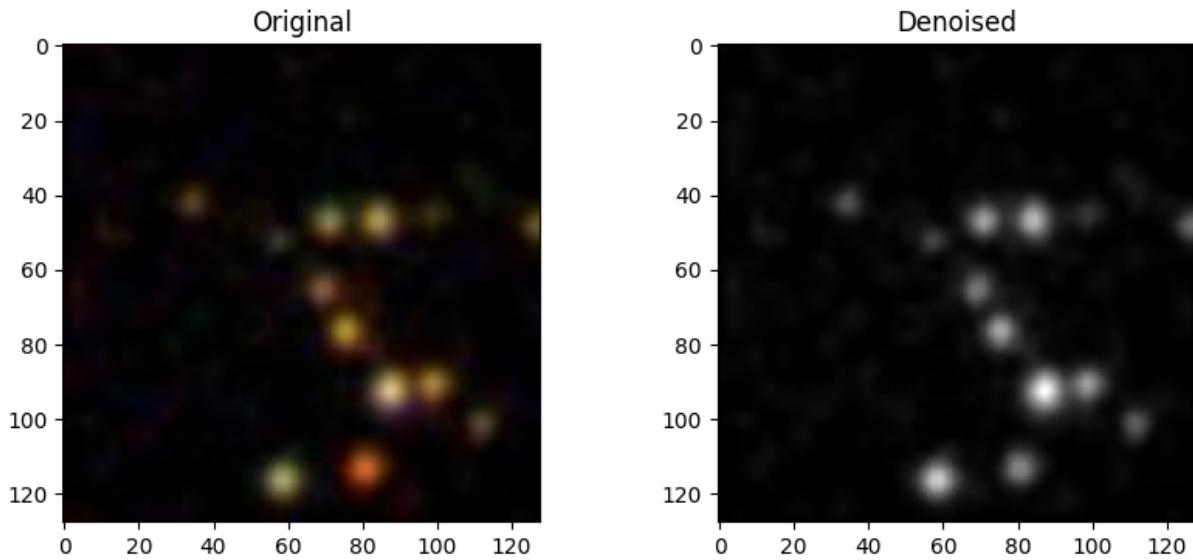
Los resultados fueron evaluados utilizando la entropía, aunque esta no es una métrica de calidad, puede ayudar a determinar si se perdió información y detalles en caso de que la entropía de la imagen original sea mayor que luego de aplicada la reducción de ruido. En el caso de que sea muy similar significa que no fue solucionada la eliminación de ruido.

#### 6.1.1. Wiener

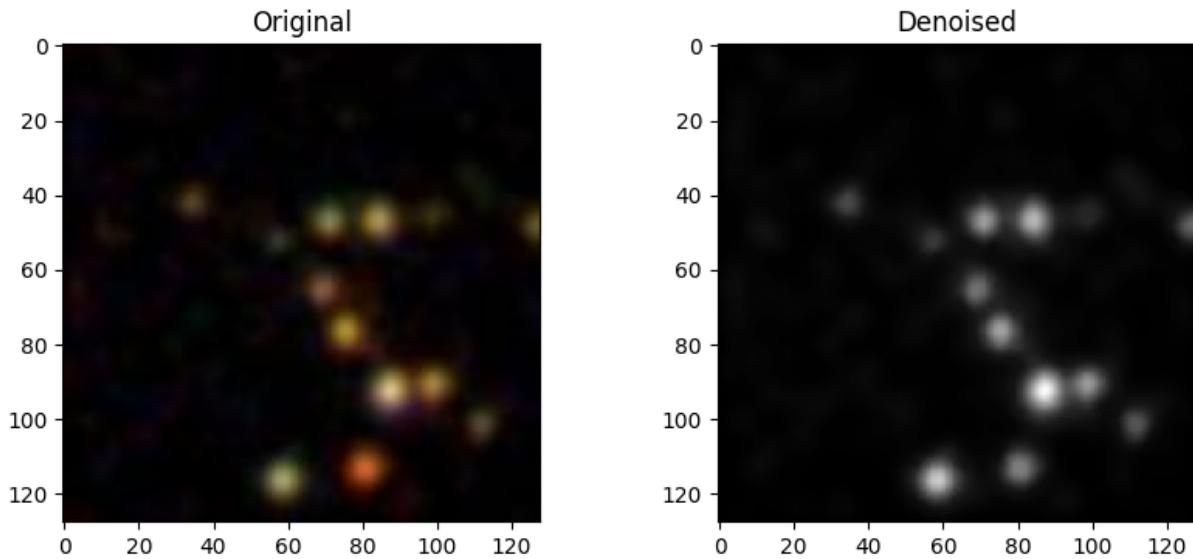
Este algoritmo podemos utilizarlo fácilmente gracias a la librería *scipy* en Python, importándolo simplemente con la siguiente línea de código.

```
|| from scipy.signal import wiener
```

Su parámetro es el kernel que tomaremos para los píxeles que tomará para realizar la corrección. Tomar un kernel más grande implicaría perder más detalles, por lo que es importante seleccionar un kernel que sea eficaz para la detección de galaxias y estrellas adecuadamente.



**Figura 15:** Wiener con kernel (3,3), Entropíaantes = 4,29, después = 4,12



**Figura 16:** Wiener con kernel (7,7), Entropíaantes = 4,29, después = 4,09

Podemos observar que el kernel (7,7) se tiene una pérdida significativa en los bordes de las estrellas, se encuentran muy borrosas y se puede seguir notar el ruido, solo que ahora se encuentra más suavizado. Mientras que con el kernel (3,3) se tienen más definidas las estrellas pero el ruido sigue estando presente. Se concluyó que no es un algoritmo tan eficaz como pensábamos para utilizar con imágenes astronómicas. Para los dos kernels, la

entropía tuvo una reducción mínima, por lo tanto su reducción de ruido no es significativa y por sus resultados se puede apreciar que no mejora la calidad de la imagen.

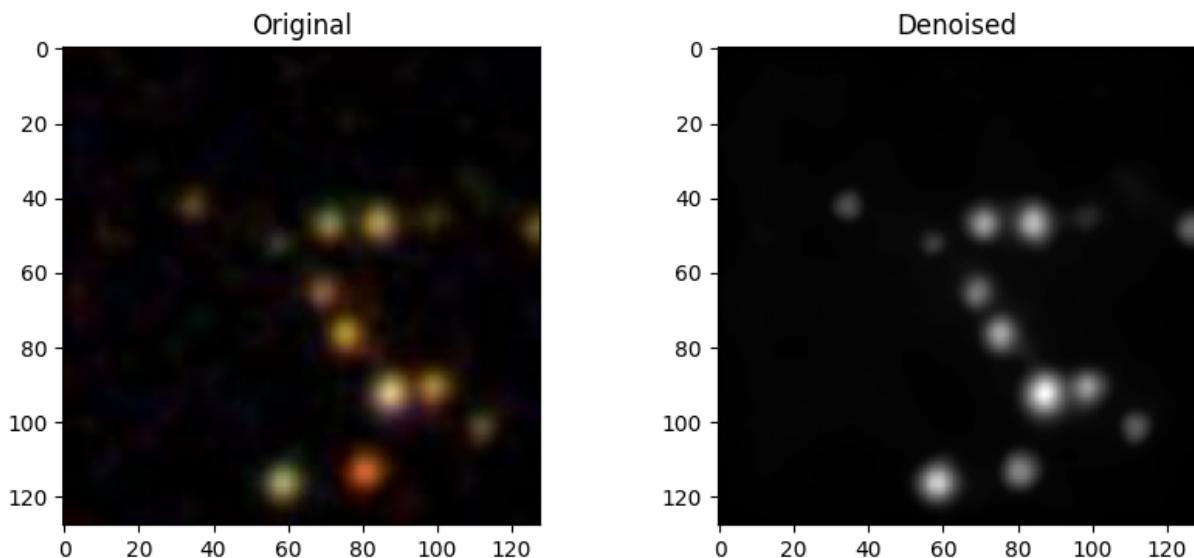
### 6.1.2. Non-Local Means

Non-Local Means presenta 3 parámetros que pueden ser modificados. Los valores elegidos para la experimentación fueron los siguientes

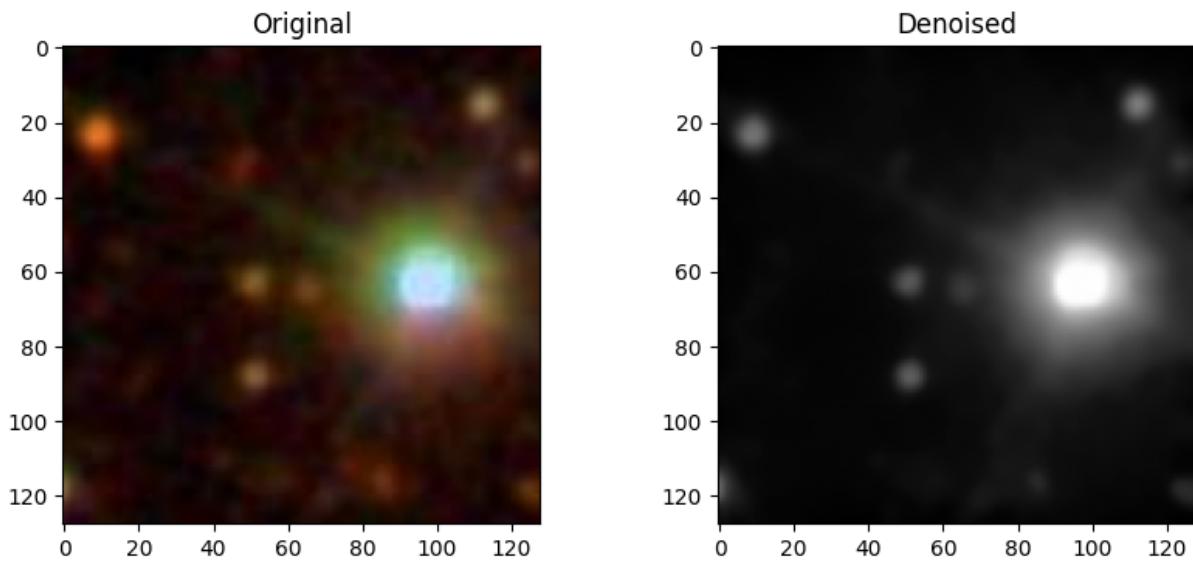
$$h = [1,5,10,15], \text{small\_window} = [5,7], \text{big\_window} = [9,21]$$

Se realizaron experimentos y los valores con los que mejor se desempeña para las imágenes utilizadas son:

$$h = 10, \text{small\_window} = 5, \text{big\_window} = 21$$



**Figura 17:** Non Local Means para estrellas con parámetros  $h = 10, \text{small\_window} = 5, \text{big\_window} = 21$ , Entropía antes = 4,18, Entropía después = 3,52

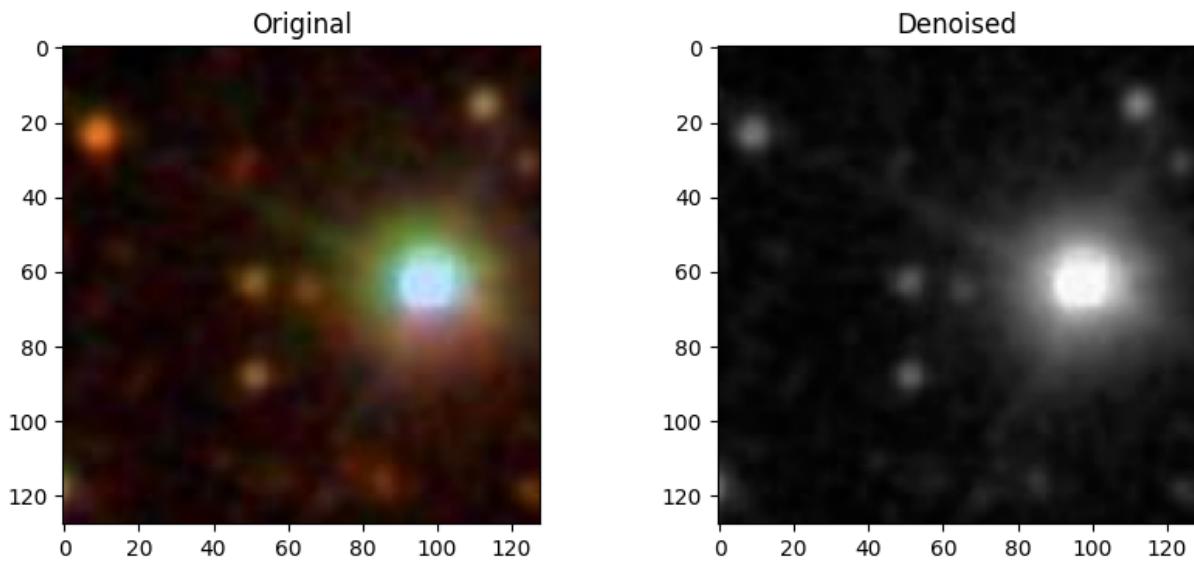


**Figura 18:** Non Local Means para galaxias con parámetros  $h = 10$ ,  $small\_window = 5$ ,  $big\_window = 21$ , Entropía antes = 6,13, Entropía = 5,65

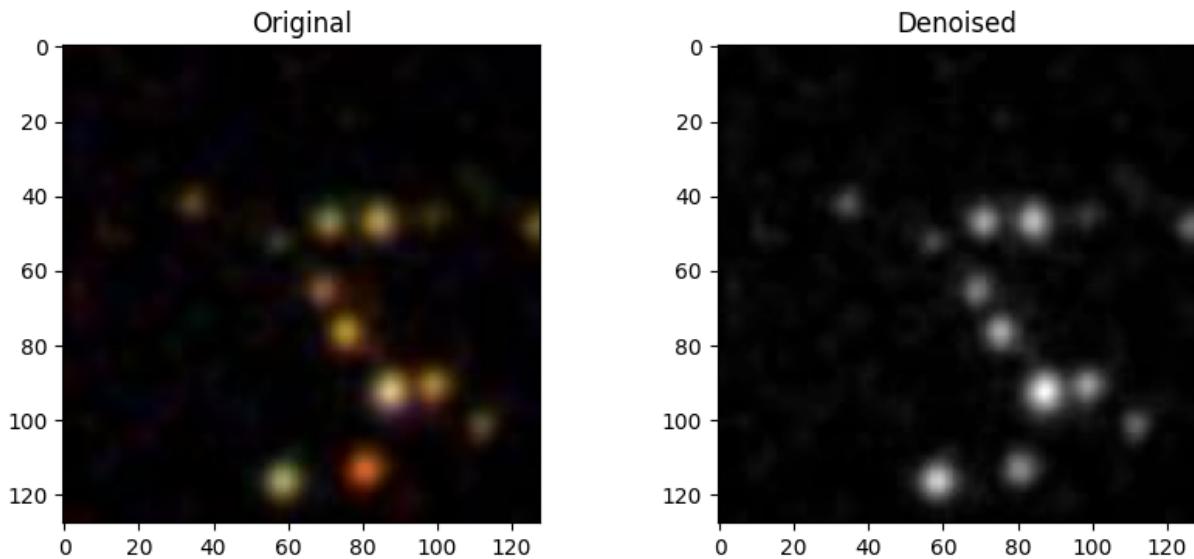
En este caso, Non-Local Means es un algoritmo que realiza una eliminación de ruido bastante eficaz, logrando quedarse con lo esencial de las imágenes y además no elimina las estrellas que se encuentran más tenues. Se puede ver que su entropía es una diferencia mayor a Wiener y se nota que el ruido se elimina exitosamente, dejando prácticamente un fondo negro con los cuerpos celestes a detectar más claros.

#### 6.1.3. Power Spectrum Analysis

El parámetro a variar es `keep_fraction`, donde para mantener más detalles pequeños para galaxias compactas, estrellas, lo mejor es aumentarlo y tenerlo entre 0.3 a 0.5 pero no se elimina el ruido completamente ya que este es el porcentaje de frecuencias que se conservan en la imagen. Una vez procesada la imagen será almacenada en una nueva ubicación junto con sus metadatos en un nuevo archivo .csv llamado “`features_step1`”, dónde almacenaremos los metadatos de todas las imágenes. A las columnas de los metadatos anteriores se suman las columnas: `psa_mean_power`, `psa_low_high_ratio` y `psa_prelim_class_guess`



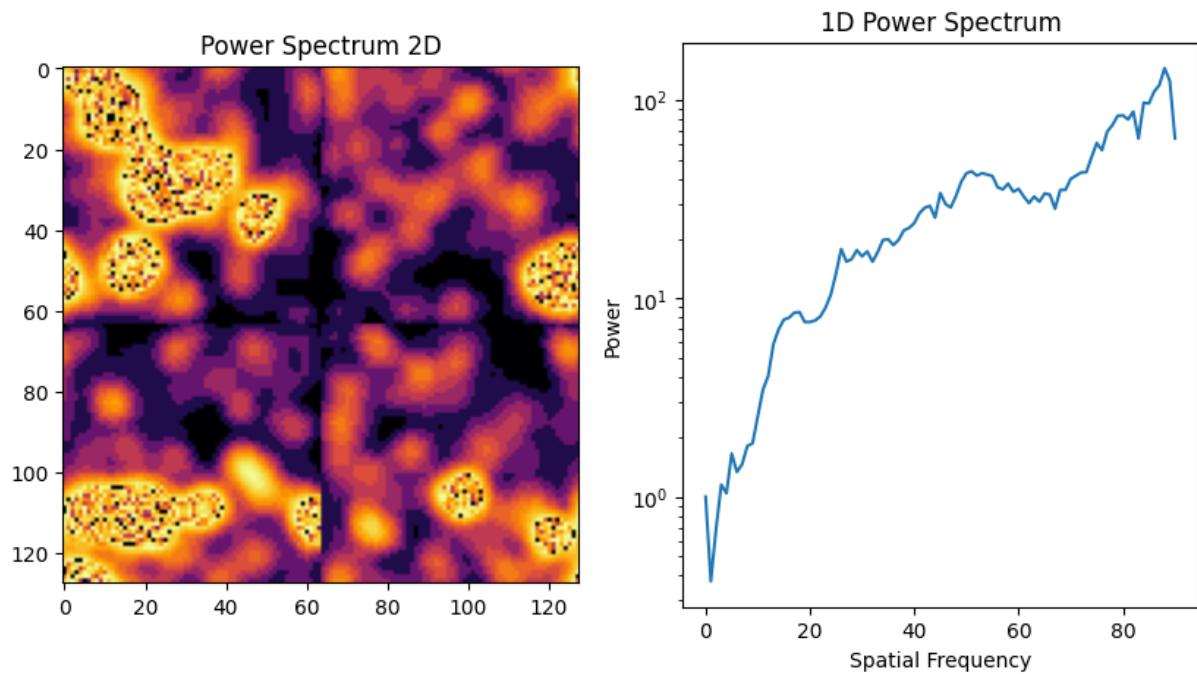
**Figura 19:** Power Spectrum Analysis para galaxias con parámetro  $keep\_fraction = 1,0$ , Entropía antes = 6,13, Entropía = 5,91



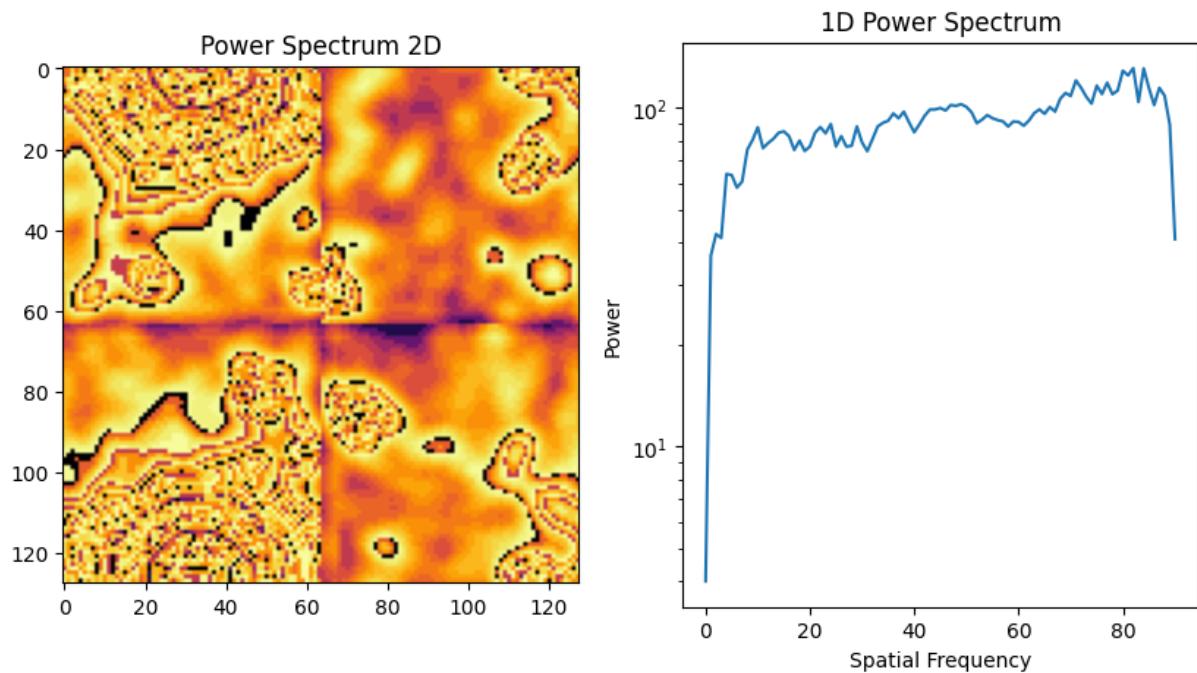
**Figura 20:** Power Spectrum Analysis para estrellas con parámetro  $keep\_fraction = 1,0$ , Entropiaantes = 4,29, Entropía : 4,20

Las imágenes no mejoraron ante ninguna variación del parámetro y la diferencia entre sus entropías antes y después es realmente mínima, y en las imágenes se logra observar que el ruido no es disminuido. De todas formas, Power Spectrum Analysis permite obtener

nuevas características para tener más información para el modelo de datos. Se obtendra el poder en función de la frecuencia espacial.



**Figura 21:** Power Spectrum Analysis para estrellas



**Figura 22:** Power Spectrum Analysis para galaxia

La primera imagen en las galaxias se nota una textura continua con estructuras curvas, es característico de fuentes extendidas como galaxias, donde hay mayor densidad de patrones suaves y amplios. Mientras que en las estrellas tenemos puntos brillantes muy definidos, lo que concuerda con fuentes puntuales como las estrellas. También se presenta un fondo oscuro, sin un patrón como en la galaxia.

Para la segunda imagen podemos ver que es relativamente plano, indicando que muchas frecuencias espaciales están presentes, especialmente medias-altas. Mientras que la imagen de estrellas sugiere predominancia de detalles finos, debido a su pendiente más marcada, aumentando hacia frecuencias altas.

## 6.2. Detección de bordes

Para diseñar una estrategia de detección de bordes eficaz y específica para imágenes astronómicas, se evaluaron cuatro algoritmos complementarios: Canny, detección de crestas multi-escala (filtro de Frangi), descomposición wavelet adaptativa y el gradiente del Laplaciano (GoL). Cada método explota propiedades distintas de la imagen, por lo que en

conjunto abarcan una amplia diversidad de estructuras astronómicas. Los parámetros se optimizaron mediante experimentación.

### 6.3. Canny

El algoritmo de Canny exige ajustar con precisión los dos umbrales que controlan la sensibilidad. Dada la amplia gama dinámica de las imágenes astronómicas, se parametrizaron los umbrales con la magnitud en banda  $z$  del SDSS ( $z$ ):

$$\text{low\_thresh} = 30 + (22 - z) \times 5, \quad (7)$$

$$\text{high\_thresh} = 80 + (22 - z) \times 10, \quad (8)$$

sujetos a los intervalos prácticos  $[10, 200]$  (bajo) y  $[30, 255]$  (alto). La selección de parámetros se basó en el  $F_1$ -score obtenido sobre cincuenta imágenes SDSS, desde estrellas brillantes ( $z < 19$ ) hasta galaxias débiles ( $z > 22$ ). Se probaron pares de umbrales de los conjuntos  $\{20, 30, 40, 50\}$  y  $\{60, 80, 100, 120\}$ ; los incrementos de 5 y 10 por magnitud mantuvieron  $F_1 > 0,70$  y controlaron el ruido.

### 6.4. Detección de crestas multi-escala (Frangi)

El filtro de Frangi es idóneo para brazos espirales y filamentos alargados. Se relacionaron las escalas gaussianas con el índice de color UV-óptico ( $u - z$ ):

$$s_{\min} = \max\left(1, \left\lfloor 1 + \frac{u - z}{2} \right\rfloor\right), \quad \text{scales} = \{s_{\min}, s_{\min} + 1, s_{\min} + 2\}.$$

Así, los objetos más azules se analizan en escalas finas y los más rojos en escalas gruesas. La sensibilidad a la curvatura ( $\beta$ ) se vinculó al color ( $g - r$ ):

$$\beta = 0,5 + \frac{g - r}{2}, \quad \beta \in [0,1, 1,5].$$

La calibración sobre 30 galaxias espirales de Galaxy mostró la mejor recuperación de filamentos ( $\text{IoU} \approx 78\%$ ) en  $\beta \simeq 0,8$ .

## 6.5. Bordes mediante wavelets adaptativos

La descomposición wavelet aísla estructuras en varias frecuencias mientras suprime ruido de Poisson. Se aplicaron tres niveles: Daubechies db4

$$T = \kappa \sigma_{\text{noise}}, \quad \sigma_{\text{noise}} = \frac{\text{MAD}}{0,6745}, \quad \kappa = \text{clip}\left(1 + \frac{5}{\text{SNR} + 1}, 1,0, 4,0\right),$$

donde la SNR se estimó a partir de la magnitud en banda  $r$ . En 20 imágenes HST/-COSMOS con colas mareales tenues se obtuvo un  $recall > 90\%$  y mínimos artefactos para  $\kappa \simeq 2\text{--}3$ .

## 6.6. Gradiente del Laplaciano (GoL)

El GoL resalta bordes sutiles y transiciones débiles:

$$\text{LoG}(x, y) = \nabla^2(G_{\sigma_{\text{LoG}}} * I), \quad \text{GoL} = \|\nabla \text{LoG}\|_2,$$

con  $\sigma_{\text{LoG}} = 2,0 \text{ px}$  (equivalente a la FWHM de  $2,5''$  del SDSS) y  $\sigma_{\text{grad}} = 1,5 \text{ px}$ . La validación se realizó sobre imágenes estándar de SDSS y JWST.

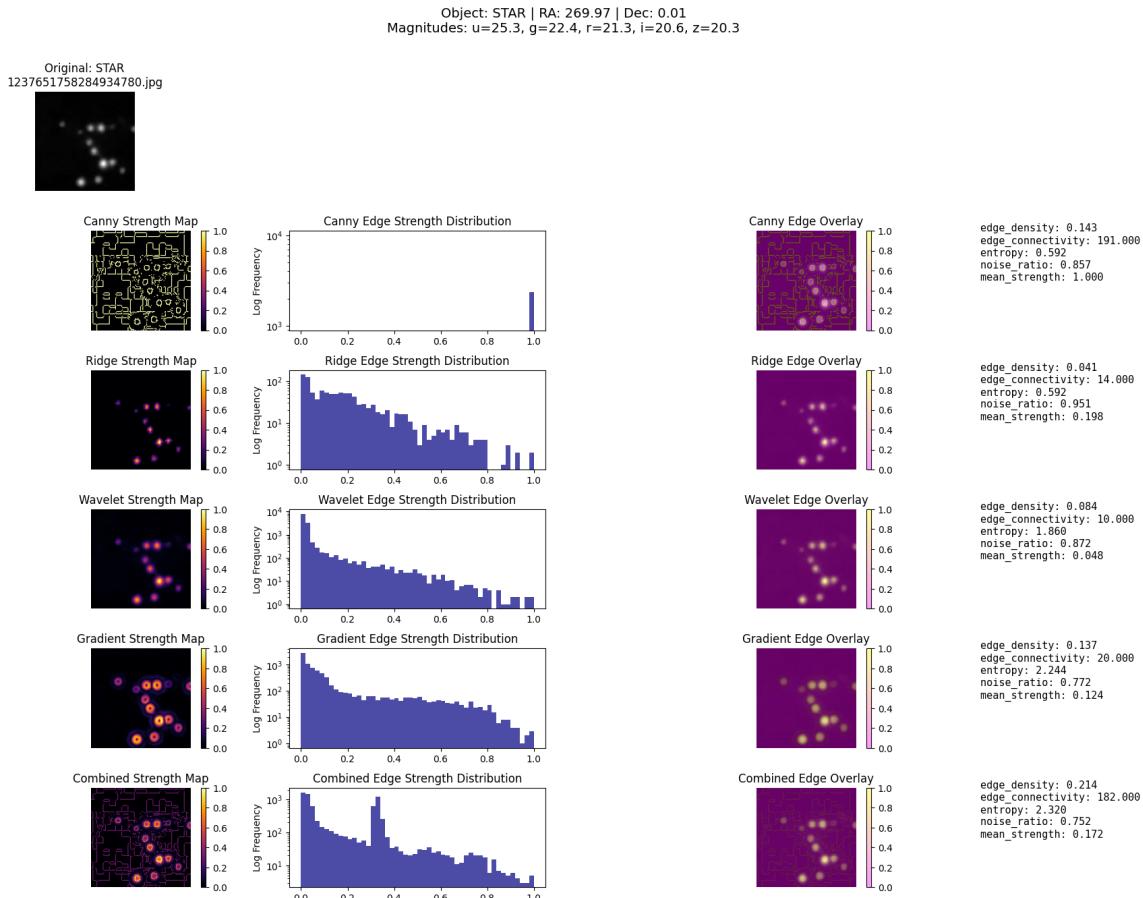
## 6.7. Comparación experimental de detectores de bordes

Se probó primero la fusión de los cuatro mapas normalizados en uno solo, extrayendo cinco estadísticas globales (densidad, conectividad, entropía, relación señal-ruido y fuerza media), pero la pérdida de información redujo su eficacia.

Por ello, se evaluó cada detector por separado y en combinación (todas las sub-conjuntos no vacíos), extrayendo en cada caso cinco rasgos y clasificando con un SVM lineal. Las precisiones obtenidas fueron:

- **Métodos individuales:** Canny 82.8 %, Frangi 84.5 %, Wavelet 84.3 %, GoL 84.8 %
- **Mejor pareja:** Wavelet + GoL 87.0 %
- **Mejor trío:** Frangi + Wavelet + GoL **87.8 %**
- **Todos los métodos:** 87.8 %

El trío Frangi–Wavelet–GoL resultó óptimo por su complementariedad: describe filamentos alargados, contornos multi-escala y bordes de bajo contraste, maximizando la precisión de clasificación. Se adoptaron, por tanto, sus 15 rasgos para el modelo final.



**Figura 23:** Detección de bordes para estrellas

## 6.8. Segmentación

### 6.8.1. Chan-Vase

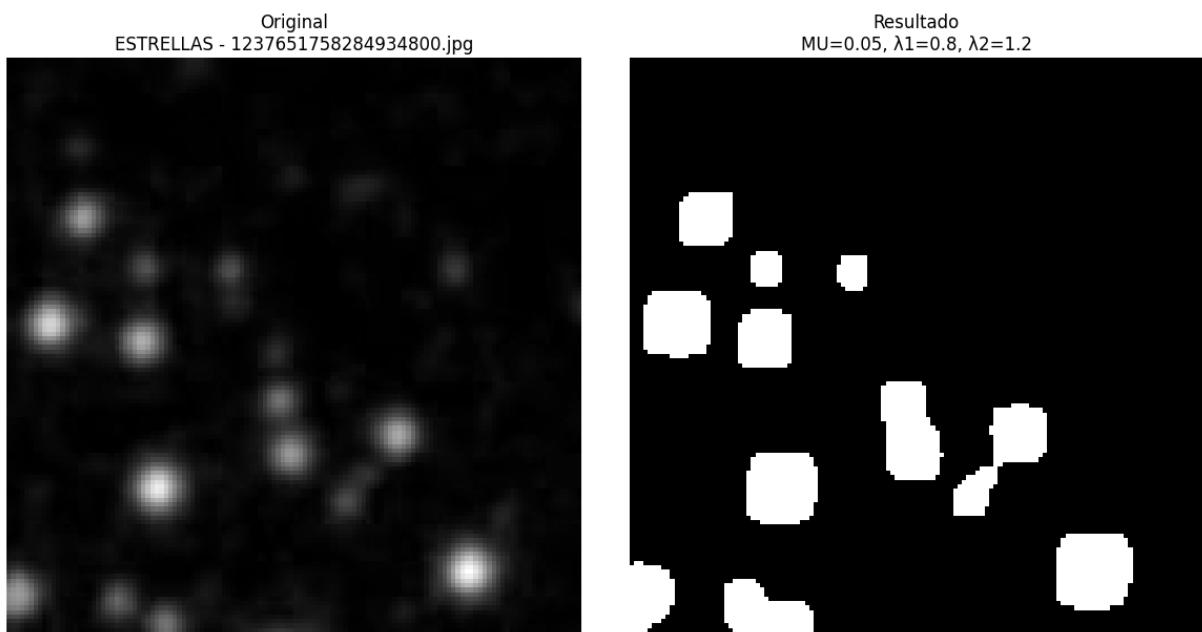
**Variación del  $\mu$**  El parámetro  $\mu$ , es uno de los valores más importantes del algoritmo. Este parámetro ajusta la penalización de longitud, que se equilibra entre ajustar la imagen de entrada con mayor precisión ( $\mu$  más pequeño) frente a producir un límite más suave ( $\mu$  mayor). Generalmente, un  $\mu$  más pequeño separará los objetos en la imagen de manera más atómica, en nuestro caso, permitiendo captar pequeñas estrellas a costa de captar una mayor cantidad de ruido mientras que un  $\mu$  mayor los separará en agrupaciones de estrellas, perdiendo información en el proceso.

**Variación del  $v$**  El parámetro  $v$  establece la penalización (o recompensa, si  $v < 0$ ) para el área dentro de  $C$ . Este parámetro es significativo sólo cuando existe una delimitación clara de dentro y fuera de la frontera de segmentación. Cuando  $v$  es demasiado negativo, el límite se expande para llenar todo el dominio, y cuando  $v$  es demasiado positivo, el límite se encoge hasta desaparecer. Para poder segmentar correctamente las estrellas, necesitaremos penalizar las áreas grandes para poder aislarlas del fondo, por lo tanto el valor debería ser  $> 0$ . Por otro lado, si queremos incluir regiones más extensas hay que establecer el valor del  $v \leq 0$

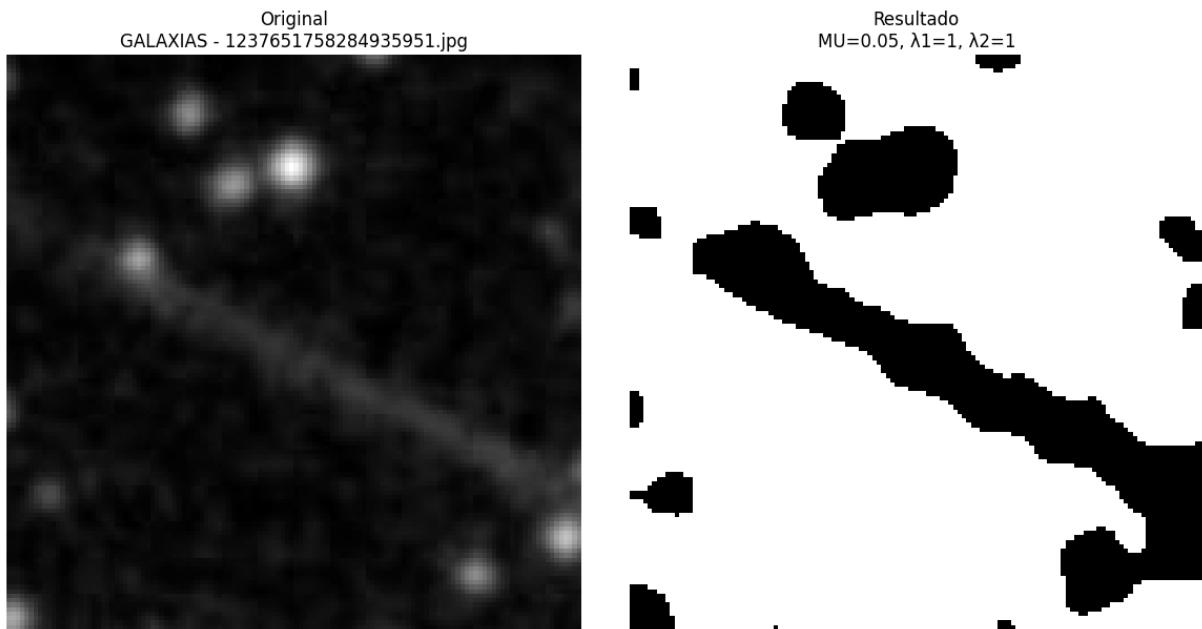
Nota: el algoritmo implementado por la librería no incluye la penalización  $v$ , por lo tanto se define  $\lambda_1$  y  $\lambda_2$  para implementar esto último. Por lo tanto,  $\lambda_1$  define el peso asignado para el contenido dentro del contorno del objeto, mientras que  $\lambda_2$  define el peso asignado para el contenido fuera del contorno del objeto.

**Inicialización y reinicialización** Durante las iteraciones la función de nivel puede perder su delimitación, lo que afecta negativamente al algoritmo, se utiliza la reinicialización permite mantener la delimitación.

Por ejemplo, para las estrellas, en una iteración antes de retornar el resultado, se pueden ver las siguientes delimitaciones:

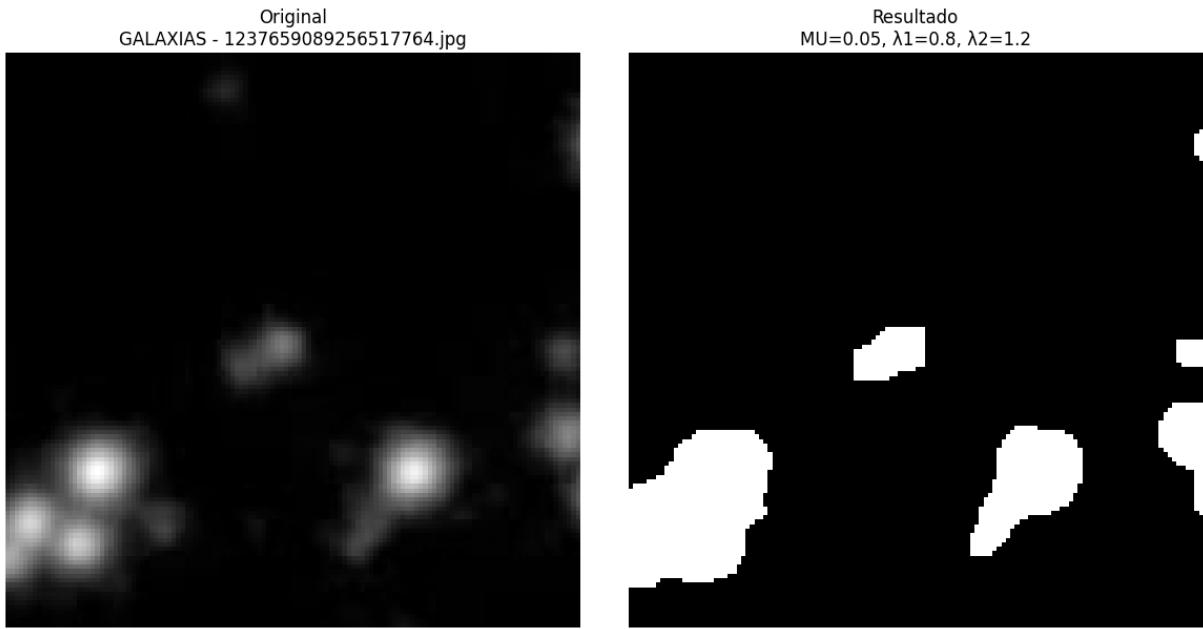


**Figura 24:** Chan - Vese aplicado en estrellas.  $\mu = 0,05 \lambda_1 = 0,8$  y  $\lambda_2 = 1,2$

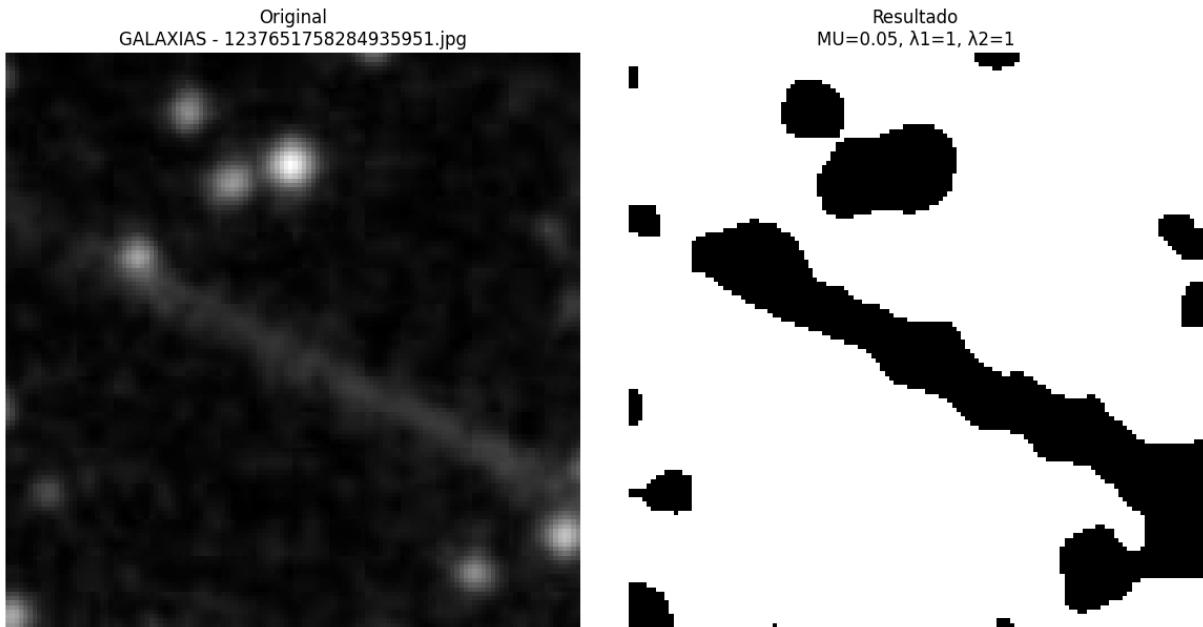


**Figura 25:** Chan - Vese aplicado en estrellas.  $\mu = 0,05 \lambda_1 = 0,8$  y  $\lambda_2 = 1,2$

Mientras que para las galaxias:



**Figura 26:** Chan - Vese aplicado en galaxias.  $\mu = 0,05$ ,  $\lambda_1 = 0,8$  y  $\lambda_2 = 1,2$



**Figura 27:** Chan - Vese aplicado en galaxias.  $\mu = 0,05$ ,  $\lambda_1 = 1$  y  $\lambda_2 = 1$

Se podía ver que al disminuir demasiado el  $\mu$ , sucedía que se capturaba demasiado ruido en la imagen (por ejemplo 0.001). Cuando se probó con aumentarlo a un número mayor,

como por ejemplo a 0.05, sucedía que no penalizaba correctamente, no logrando delimitar correctamente las estrellas y galaxias. Por lo tanto llego a la conclusion que el mejor valor era 0.01.

Mientras que para los valores de  $\lambda_1$  y  $\lambda_2$ , debemos hacer una distinción en casos para su comparación, ya que hay que tener en cuenta diferentes posibilidades:

Primero, cuando ambos eran iguales a 1, el algoritmo no es capaz de distinguir interior de exterior de la imagen, generando que sólo se vean máximos locales de intensidad sin ninguna conexión.

Cuando  $\lambda_1 > 1$  y  $\lambda_2 = 1$  la penalización interna era muy alta, de forma que las estrellas pequeñas se mezclaban con el fondo y no se segmentaban correctamente.

Cuando  $\lambda_2$  era más pequeño que  $\lambda_1$  (ejemplo  $\lambda_1 = 1$  y  $\lambda_2 = 0,5$ ), no se aplica la suficiente penalización en el fondo de la imagen, lo que genera que se pierdan estrellas / galaxias tenues.

Por último cuando  $\lambda_1 = 1$  y  $\lambda_2 > 1,2$  sucede que la penalización externa es demasiado grande, haciendo que el tamaño de las estrellas / galaxias aumente drásticamente, perdiendo su forma.

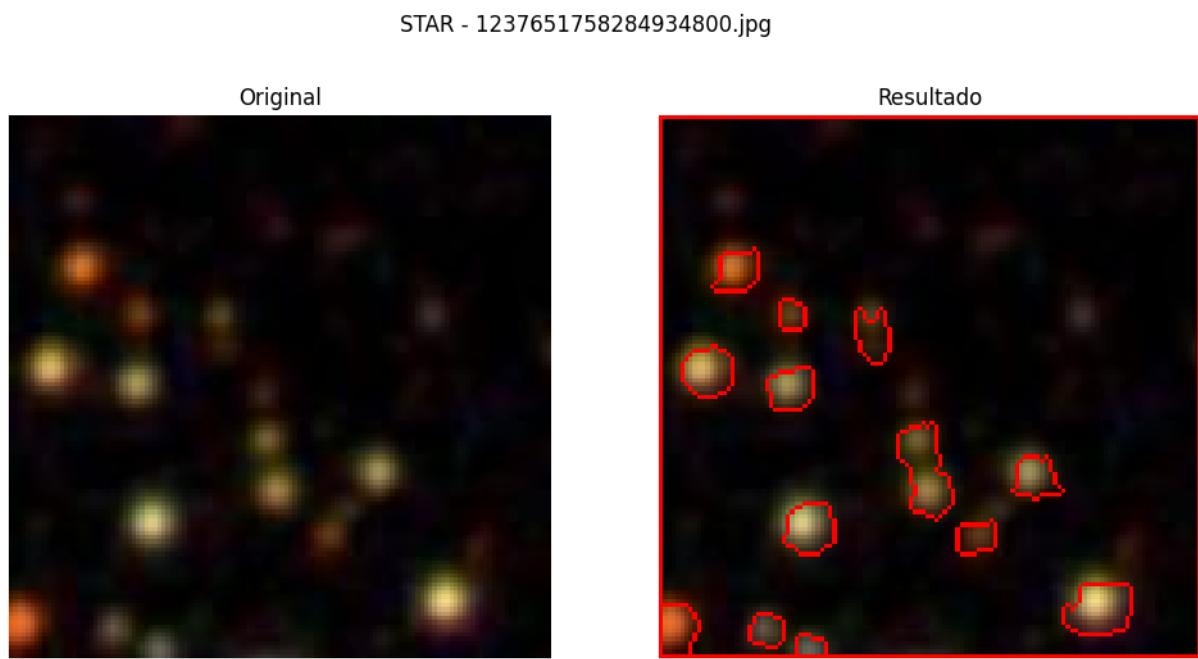
Concluimos que, los mejores valores son:

$$\lambda_1 = 0,5, \quad \lambda_2 = 1$$

Ya que,  $\lambda_1 = 0,5$  reduce la sensibilidad al ruido interno, conservando sólo estrellas / galaxias reales.  $\lambda_2 = 1$  refuerza la detección de halos difusos, extendiendo el contorno sobre las estrellas / galaxias. Aunque también se noto que en algunos casos, estos valores generaban un poco de ruido en algunos pixeles aislados

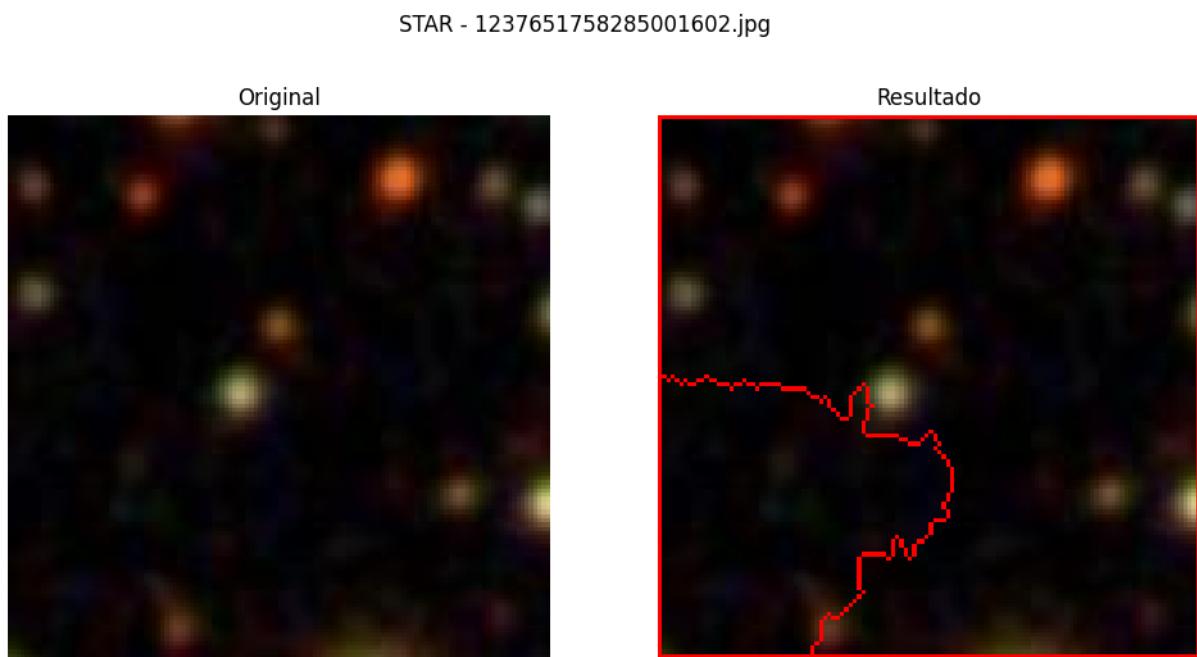
### 6.8.2. Watershed

Todas las imagenes son el retorno del algoritmo.



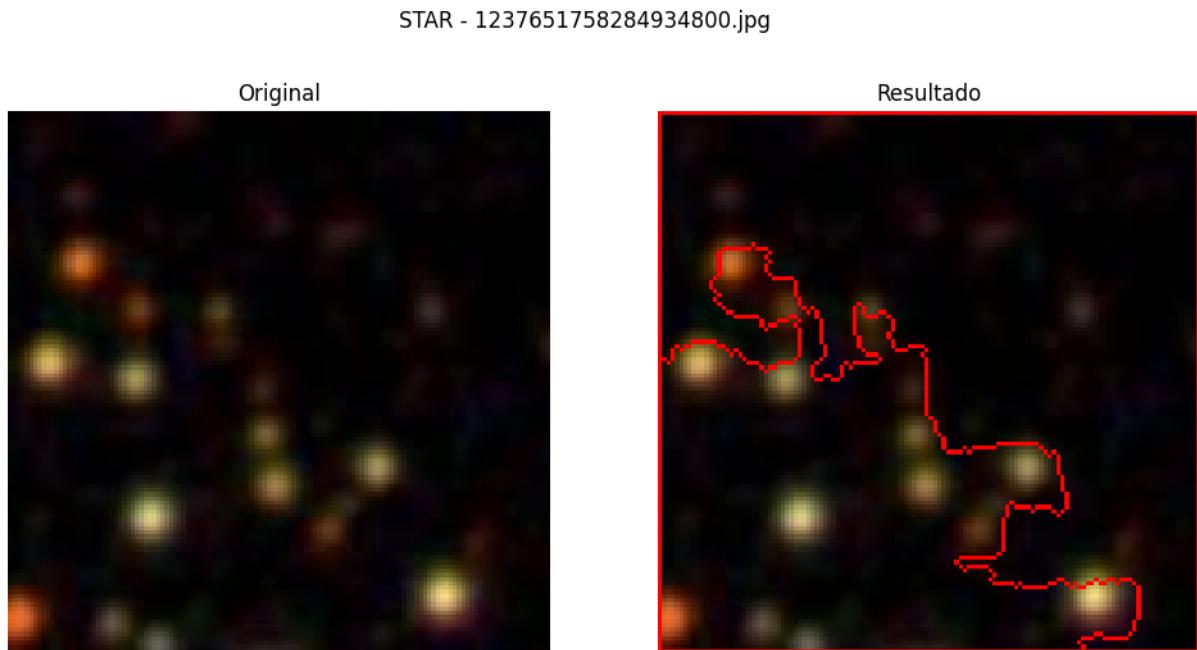
**Figura 28:** Watershed con multiplicador de threshold 0,9

**Multiplicador del threshold** Se eligio el valor de 0,9, ya que en valores menores el algoritmo tomaba secciones donde no existían objetos, mientras que con valores mayores el algoritmo agrupa demasiados objetos sin segmentarlos correctamente.



**Figura 29:** Watershed con threshold mayor a 0,9

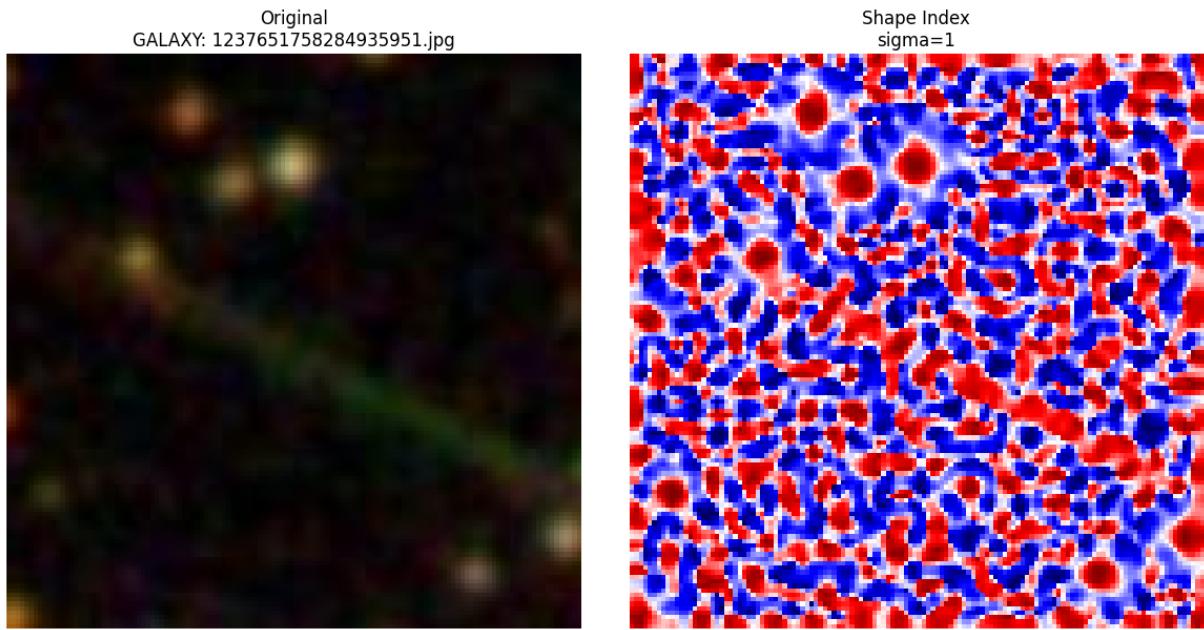
**open\_iter:** Se eligio el valor de 1, ya que valores mayores suavizaba demasiado la imagen, perdiendo información de la misma.



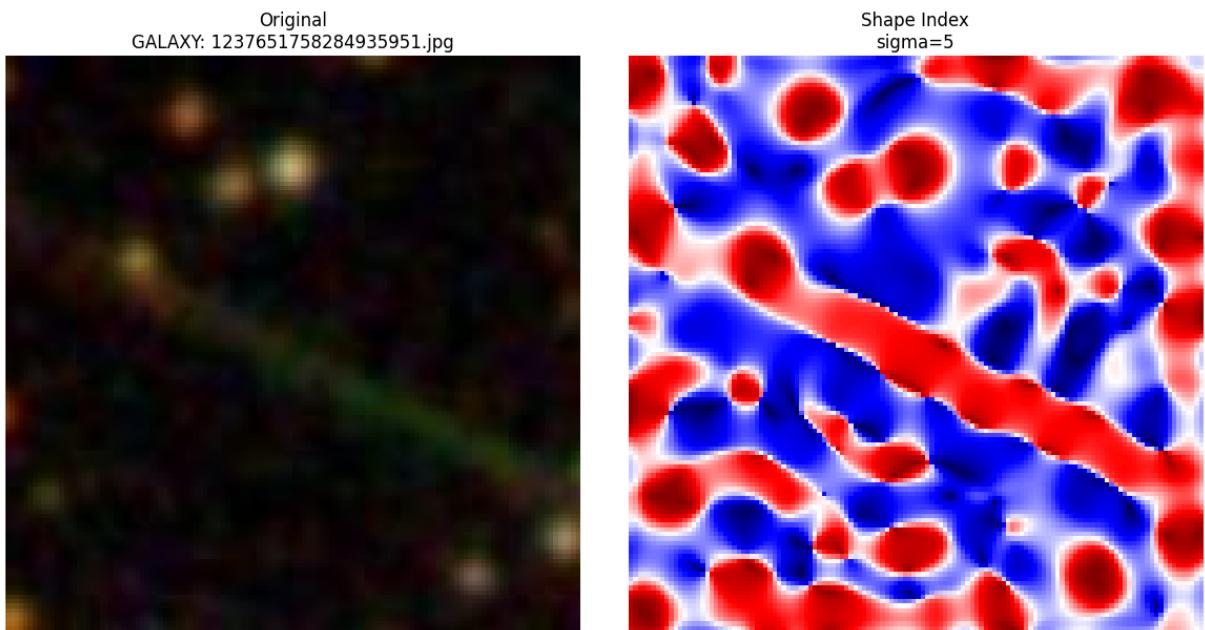
**Figura 30:** Watershed con iteraciones mayores a 1

### 6.8.3. Shape Index

Como se menciono anteriormente, esta técnica no tiene parámetros obligatorios. Sin embargo, se noto que la variación del  $\sigma$ . brindaba beneficios a esta tecnica. El valor elegido fue 3, ya que cuando el valor era menor, la técnica detectaba formas donde no existían y para valores mayores se perdía demasiada información.



**Figura 31:** Shape index con  $\sigma < 3$



**Figura 32:** Shape index con  $\sigma > 3$

## 6.9. Clasificación

**Diseño experimental.** Se evaluaron varios algoritmos supervisados con el fin de seleccionar el clasificador que mejor generalizara sobre el conjunto de imágenes astronómicas. El *dataset* (2000 objetos) se dividió en 80 % para entrenamiento, 10 % para validación y 10 % para prueba. Todos los hiperparámetros se optimizaron mediante *grid-search* con validación cruzada estratificada de cinco pliegues; la métrica principal fue la *accuracy* y se registraron también *precision*, *recall* y *F1*. Las variables de entrada proceden de la etapa de procesamiento de imágenes (fotometría, espectro de potencia, medidas de bordes y segmentación). Para estudiar la influencia de la dimensionalidad se repitió el experimento aplicando *PCA* hasta retener el 95 % de la varianza.

### Modelos considerados.

- **SVM con núcleo RBF** [14]:  $C \in \{0,1,1,10\}$ ,  $\gamma \in \{0,01,0,1,\text{scale}\}$ .
- **Random Forest** [11]:  $n_{\text{árboles}} \in \{100, 200\}$ .
- **Gradient Boosting** [12]:  $n_{\text{estim}} = 100$ ,  $\text{learning-rate} \in \{0,05,0,1\}$ .
- **XGBoost** [15]:  $\eta = 0,1$ , profundidad = 6,  $\text{subsample} = 0,8$ .

**Resultados.** La Tabla 4 sintetiza la exactitud promedio en la partición de validación (400 imágenes).

**Tabla 4:** Exactitud de los clasificadores en validación.

Modelo	Accuracy
SVM–RBF	0.74
Random Forest	0.77
Gradient Boosting	0.77
XGBoost	0.75

## Discusión

1. **Gradient Boosting obtiene la mayor exactitud** (84.1 %), superando a la SVM en 4 puntos porcentuales y mostrando mejor equilibrio *precision–recall* entre clases.
2. **El PCA no aporta ventaja.** La ligera caída al usar componentes principales indica que las variables originales ya están razonablemente des-correlacionadas y contienen la información discriminativa necesaria.
3. **XGBoost se aproxima**, pero su ganancia frente a Gradient Boosting no compensa el coste computacional adicional dado el tamaño del *dataset*.

**Modelo adoptado.** Se elige **Gradient Boosting** con  $n_{\text{estim}} = 100$  y  $\text{learning-rate} = 0,1$  como clasificador final del *pipeline*, al ofrecer la mejor exactitud validada y un compromiso favorable entre rendimiento y complejidad de entrenamiento.

## 7. Manual de usuario

El mismo se puede encontrar en el Readme del repositorio de [Github](#)

## 8. Conclusiones

En este proyecto se investigó la posibilidad de clasificar imágenes astronómicas de forma de poder identificarlas como galaxias o estrellas, basándonos en el paper elegido.

La investigación, desarrollo, implementación y documentación fue todo un desafío en cada una de sus secciones para todos nosotros, debido a que para resolver el problema planteado debíamos explorar y comprender algoritmos abstractos con una gran complejidad matemática.

Una vez que teníamos seleccionado los algoritmos, tuvimos que encontrar un dataset de imágenes válido para efectuar el análisis.

Una vez obtenidos los datos y teniendo los algoritmos implementados, debíamos encontrar parámetros preliminares para cada uno en base a la comprensión del mismo y la experimentación, para luego añadir todos los algoritmos al pipeline, el cual desarrollamos nosotros mismos, volviendo a tunear los parámetros en el momento para encontrar los mejores valores.

Por último, la elección del modelo tampoco fue sencillo, debido a que cada modelo precisaba también una gran cantidad de experimentación. Todo esto también implicaba una inversión de tiempo, debido a que ejecutábamos los algoritmos de manera local y un error puede llevar a tener que ejecutar el pipeline nuevamente.

Luego de toda la experimentación y la implementación del pipeline, consideramos que logramos cumplir el objetivo del proyecto ampliamente, ya que el resultado del modelo es considerablemente bueno.

Aunque estamos muy satisfechos con los resultados obtenidos luego de nuestro trabajo, consideramos que quizás fue muy ambicioso intentar implementar un pipeline tan complejo, esto no solo se ve en la extensión del código sino que también en la extensión de este documento.

Queremos también destacar que en este trabajo también se puede notar que el análisis de imágenes astronómicas es un campo que sigue en crecimiento y que podemos esperar una gran evolución en los próximos años en el mismo.

## 9. Autoevaluación de cada miembro

### 9.1. Bruno Quadrelli

<b>Categoría</b>	<b>Subcategoría</b>	<b>Puntuación</b>			
		Excelente (1 pto)	Bien (0,65 pto)	Regular (0,35 pto)	Mal (0 pto)
<b>Exposición</b>	Comprendión y dominio				
	Exposición didáctica				
	Integración del equipo				
<b>Implementación</b>	Objetivos	x			
	Aspectos didácticos		x		
	Experimentación y conclusiones	x			
<b>Documentación</b>	Contenidos	x			
	Divulgación de contenidos	x			
	Bibliografía y recursos		x		

## 9.2. Franco Ribarov

Categoría	Subcategoría	Puntuación			
		Excelente (1 pto)	Bien (0,65 pto)	Regular (0,35 pto)	Mal (0 pto)
<b>Exposición</b>	Comprendión y dominio	x			
	Exposición didáctica			x	
	Integración del equipo	x			
<b>Implementación</b>	Objetivos	x			
	Aspectos didácticos			x	
	Experimentación y conclusiones	x			
<b>Documentación</b>	Contenidos	x			
	Divulgación de contenidos	x			
	Bibliografía y recursos	x			

### 9.3. Emmanouil-Angelos Tsigkas

Categoría	Subcategoría	Puntuación			
		Excelente (1 pto)	Bien (0,65 pto)	Regular (0,35 pto)	Mal (0 pto)
<b>Exposición</b>	Comprendión y dominio	x			
	Exposición didáctica		x		
	Integración del equipo	x			
<b>Implementación</b>	Objetivos	x			
	Aspectos didácticos		x		
	Experimentación y conclusiones	x			
<b>Documentación</b>	Contenidos	x			
	Divulgación de contenidos	x			
	Bibliografía y recursos	x			

## 10. Tabla de tiempos

### Registro de Horas - Angel

Date	Phase	Activity	Hours
12/2/2025	1	Search project topic	2
24/2/2025	1	Idea Documentation	1
24/2/2025	1	Literature review	1
8/3/2025	2	Research edge detection approaches	2
10/3/2025	2	Implement Canny & Laplacian-of-Gaussian	3

10/3/2025	2	Research & implement ridge and wavelet detectors	4	
11/3/2025	2	Parameter tuning & algorithm combination	3	
11/3/2025	2	Initial segmentation example & documentation	2	
13/3/2025	2	Data cleaning; drop irrelevant columns & research handling methods	3	
13/3/2025	2	Implement classification algorithms (SVM, RF, etc.)	3	
2/4/2025	3	Generate confusion matrices & classification testing	2	
2/4/2025	3	Add models: Gradient Boosting & XGBoost	3	
4/4/2025	3	Compare classifiers; accuracy vs. recall analysis	2	
4/4/2025	3	Documentation of classification experiments	2	
7/4/2025	4	Test PCA for dimensionality reduction	1	
12/4/2025	4	Tune adaptive Canny thresholds based on brightness	3	
13/4/2025	4	Calibrate Frangi filter parameters using color metadata	2	
14/4/2025	4	Set wavelet thresholds via r-band SNR	2	
19/4/2025	4	Configure Laplacian gradient for telescope-induced blur	2	
21/4/2025	4	Integrate final edge detection into pipeline & refactor code	4	
22/4/2025	4	Test Classification on Unknown Images	3	
25/4/2025	4	Rewriting of the whole notebook/ Merging Notebooks	6	
29/4/2025	4	Document methodology for Phases 3-4 and activities mentioned above	10	
5/5/2025	5	Changes to edge detection documentation and theory addition	1	
8/5/2025	5	Worked on project presentation	2	
10/5/2025	5	Final changes to code and GitHub updates	1	

## Registro de Horas - Bruno

Fecha	Fase	Actividad	Horas
12/2/2025	1	Buscar Tema de Proyecto	2
21/2/2025	1	Documentación e investigación para seguimiento	1.5
22/2/2025	1	Estudio de fuente de datos y documentación	1
8/3/2025	2	Implementación de algoritmos de reducción de ruido	3
11/3/2025	2	Implementación algoritmo PSA y Wiener, documentación	5.5
2/4/2025	3	Revisión de código base	2
3/4/2025	3	Ejecución de código base	3
6/4/2025	3	Corrección código	3
7/4/2025	3	Ejecución, corrección de código y modelado ML	5
8/4/2025	3	Modelado ML y documentación	4
28/4/2025	4	Documentación resumen, introducción	2
28/4/2025	4	Planteo teórico Non Local Means y PSA	2
29/4/2025	4	Planteo teórico de detección de bordes, experimentación de eliminación de ruido, ejecución de algoritmos y detección de bordes (con Angel)	7
9/5/2025	4	Escritura de nueva introducción y desarrollo de conexión teórica con modelo ML. Ejecución del proyecto	4
10/5/2025	4	Ejecución y ajustes del notebook	3
12/5/2025	4	Arreglos en segmentación y diseño presentación, contenido en documentación	8

## Registro de Horas - Franco

Fecha	Fase	Actividad	Horas
12/2/2025	1	Buscar Tema de Proyecto	2

22/2/2025	1	Evaluación de fuente de datos (testeo de obtención de imágenes)	1
22/2/2025	1	Documentación	1
8/3/2025	2	Algoritmo de Chan Vese	3
9/3/2025	2	Algoritmo de Watershed	4
10/3/2025	2	Documentación Watershed	1.5
10/3/2025	2	Shape Index	1
11/3/2025	2	Finalización de Shape Index. Documentación Fase 2	4
4/4/2025	3	Implementación de algoritmos	3
7/4/2025	3	Unión de código, Bugfix, entrenamiento	5
7/4/2025	3	Documentación	1
8/4/2025	3	Documentación	1
28/4/2025	4	Planteamiento teórico Segmentación y Detec- ción de ruido	5.5
29/4/2025	4	Experimentación e Implementación Segmen- tación	6
11/5/2025	4	Documentación final y slides	5
12/5/2025	4	Documentación final y slides	8

## Referencias

- [1] A. Buades, B. Coll, and J.-M. Morel, “A non-local algorithm for image denoising,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 60–65, 2005.
- [2] SciPy Developers, “Non-local means denoising — scipy.signal,” 2024. [Online]. Available: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.wiener.html>
- [3] D. R. Matthys, “Lab 02: Fourier analysis of images,” 2002. [Online]. Available: <https://academic.mu.edu/phys/matthysd/web226/Lab02.htm>
- [4] A. Addison, “How the laplacian of gaussian filter works,” 2020. [Online]. Available: <https://automaticaddison.com/how-the-laplacian-of-gaussian-filter-works/>

- [5] OpenCV Documentation, “Canny edge detection in opencv,” 2021, accessed: 2025-04-30. [Online]. Available: [https://docs.opencv.org/4.x/da/d22/tutorial\\_py\\_canny.html](https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html)
- [6] Roboflow, “Edge detection: An introduction,” 2021, accessed: 2025-04-30. [Online]. Available: <https://blog.roboflow.com/edge-detection/>
- [7] scikit-image contributors, “Ridge filter example — scikit-image documentation,” 2023. [Online]. Available: [https://scikit-image.org/docs/0.25.x/auto\\_examples/edges/plot\\_ridge\\_filter.html](https://scikit-image.org/docs/0.25.x/auto_examples/edges/plot_ridge_filter.html)
- [8] T. Lindeberg, “Edge detection and ridge detection with automatic scale selection,” 1998, vol. 30, no. 2, pp. 117–154.
- [9] B. Cui and H. Jiang, “An image edge detection method based on haar wavelet transform,” 2017, undergraduate Research Paper.
- [10] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [11] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [12] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.
- [13] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.
- [14] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “A training algorithm for optimal margin classifiers,” in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT)*, 1992, pp. 144–152.
- [15] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.