In [ ]:
```
# PROBLEMA 1
# Apartado 1
# foco primario
aperture = 4.0  # meters
f_ratio = 2.7
distance = 10.0  # meters
focal_length = aperture * f_ratio
scale = 206265 / focal_length

print('La longitud focal del telescopio Mayall para su foco primario es:',fo
print('La escala de imagen del telescopio Mayall para su foco primario es:',

# foco Cassegrain
f_ratio = 8.0
focal_length = aperture * f_ratio
scale = 206265 / focal_length

print('La longitud focal del telescopio Mayall para su foco Cassegrain es:',
print('La escala de imagen del telescopio Mayall para su foco Cassegrain es:

# foco Coudé
f_ratio = 160.0
focal_length = aperture * f_ratio
scale = 206265 / focal_length

print('La longitud focal del telescopio Mayall para su foco Coudé es:',focal
print('La escala de imagen del telescopio Mayall para su foco Coudé es:',sca

# Apartado 2
aperture = 10.4  # meters
f_ratio = 16.33
distance = 10.0  # meters
focal_length = aperture * f_ratio
scale = 206265 / focal_length

print('La longitud focal del telescopio Gran Telescopio Canarias para su foc
print('La escala de imagen del telescopio Gran Telescopio Canarias para su f
```

```
La longitud focal del telescopio Mayall para su foco primario es: 10.8 m
La escala de imagen del telescopio Mayall para su foco primario es: 19098.61
111111111 arcsen/mm
La longitud focal del telescopio Mayall para su foco Cassegrain es: 32.0 m
La escala de imagen del telescopio Mayall para su foco Cassegrain es: 6445.7
8125 arcsen/mm
La longitud focal del telescopio Mayall para su foco Coudé es: 640.0 m
La escala de imagen del telescopio Mayall para su foco Coudé es: 322.2890625
arcsen/mm
La longitud focal del telescopio Gran Telescopio Canarias para su foco prima
rio es: 169.832 m
La escala de imagen del telescopio Gran Telescopio Canarias para su foco pri
mario es: 1214.5237646615478 arcsen/mm
```

In [ ]:
```
#Ejercicio 2
#Suppose that the human vision is limited from the difracction to a waveleng
1
```

Loading [MathJax]/extensions/Safe.js

```
#Calculate the angular resolution of the human eye.
lambda1=5000*u.AA
D=8*u.mm
lambda1=lambda1.to(u.m)
D=D.to(u.m)
theta=((1.22*u.rad)*lambda1/D)
theta_rad=theta.to(u.arcsec)
print("La resolución angular del ojo humano es",theta_rad)

#Apartado 2
#Compare with the maximum angular size of Venus and Jupiter as seen from the
SolVenus=0.72*u.AU
SolJupiter=5.20*u.AU
SolTierra=1*u.AU
dtv=(SolTierra-SolVenus).to(u.km)
dtj=(-SolTierra+SolJupiter).to(u.km)
radio_v=6052*u.km
radio_j=69911*u.km
th_v=2*(radio_v/dtv)*u.rad
th_j=2*(radio_j/dtj)*u.rad
print("El tamaño angular de Venus es",th_v.to(u.arcsec))
print("El tamaño angular de Jupiter es",th_j.to(u.arcsec))

# Compara con la resolución angular del ojo humano
if th_v<theta_rad:
    print("Venus no puede ser observado por el ojo humano")
else:
    print("Venus puede ser observado por el ojo humano")
if th_j<theta_rad:
    print("Jupiter no puede ser observado por el ojo humano")
else:
    print("Jupiter puede ser observado por el ojo humano")
```

```
La resolución angular del ojo humano es 15.7276914763411 arcsec
El tamaño angular de Venus es 59.60334106582341 arcsec
El tamaño angular de Jupiter es 45.90140093911414 arcsec
Venus puede ser observado por el ojo humano
Jupiter puede ser observado por el ojo humano
```

In [ ]:
```
# PROBLEMA 3
D = 300 # mm
# la longitud de onde visible es  de 550 nm
longitud_onda = 550 * 10**-9
resolucion = 1.22 * longitud_onda / D
print("El tamaño del cráter más pequeño es:", resolucion, "m")
```

```
El tamaño del cráter más pequeño es: 2.2366666666666666e-09 m
```

In [ ]:
```
import math
# PROBLEMA 4
# Variables
radio_anillos = 66000 * 1000  # Convertir de km a metros
distancia_saturno = 1.429e12  # Distancia promedio entre la Tierra y Saturno
longitud_onda = 5.50e-7  # Longitud de onda de la luz utilizada en metros

# Cálcul de la apertura mínima del telescopio
```

Loading [MathJax]/extensions/Safe.js

```
resolucion_angular = 1.22 * (longitud_onda / radio_anillos)

# Imprimir el resultado
print("La apertura mínima del telescopio necesaria para ver los anillos de S
```

```
La apertura mínima del telescopio necesaria para ver los anillos de Saturno
es: 1.0166666666666666e-14 metros
```

In [ ]:
```
import math
# PROBLEMA 5
t1 = 20  # minutes
D1 = 2.4  # meters
θ1 = 1   # arcseconds

D2 = 10.0  # meters
θ2 = 0.4   # arcseconds
S_N = 100

t2 = (D2/D1)**2 * (θ1/θ2)**2 * t1 * (S_N)**2

t2 = math.ceil(t2)  # Round up to the nearest minute

print("El tiempo que necesitaría el telescopio Keck es aproximadamente:", t2
```

```
El tiempo que necesitaría el telescopio Keck es aproximadamente: 21701389 mi
nutos.
```

In [ ]:
```
# PROBLEMA 6
# Apartado 1 y 3

from photutils.datasets import make_random_models_table
import numpy as np
import matplotlib.pyplot as plt
from photutils.datasets import make_gaussian_sources_image
from photutils.datasets import make_noise_image


n_sources = 15
param_ranges = {'amplitude': [600, 1000], # Amplitud de las fuentes
                'x_mean': [0, 500], # Posición en x de las fuentes
                'y_mean': [0, 500], # Posición en y de las fuentes
                'x_stddev': [4, 5], # Sigma en x de las fuentes
                'y_stddev': [4, 5], # Sigma en y de las fuentes
                'theta': [0, np.pi]} # Ángulo de rotación de las fuentes

sources = make_random_models_table(n_sources, param_ranges, seed=20240325)
print(sources)
# Redefinimos el tamaño de la imagen
npix_x=500
npix_y=500
shape = (npix_x, npix_y)
# Creamos la imagen con las fuentes
image = make_gaussian_sources_image(shape, sources)
# Muestra la imagen con las fuentes aleatorias
plt.figure()
plt.imshow(image, cmap='gray')
```

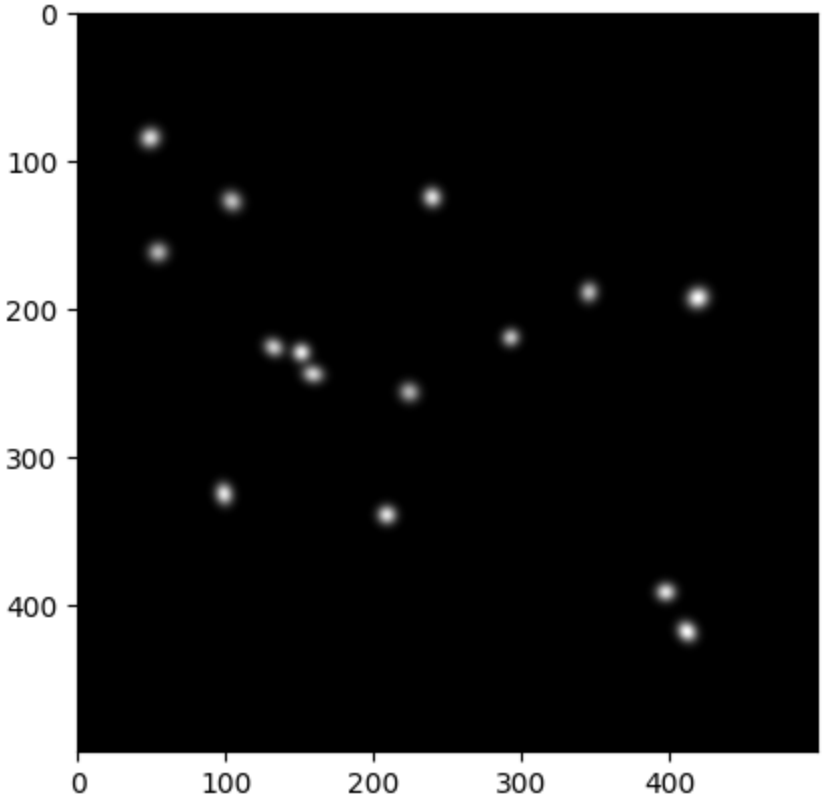
Loading [MathJax]/extensions/Safe.js

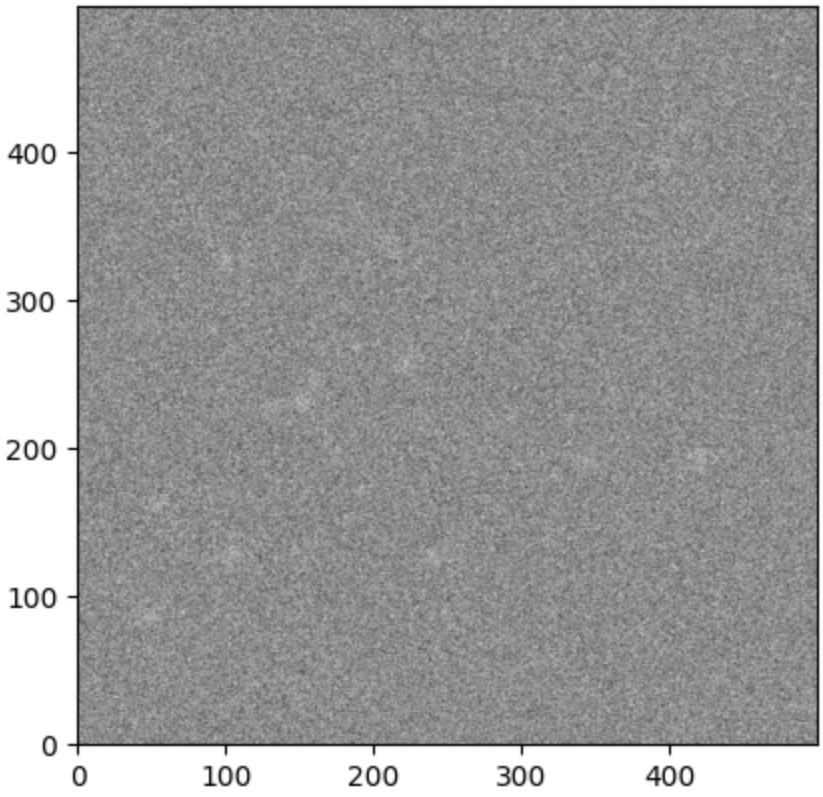
```
# Apartado 2 y 3
noise = make_noise_image(shape, distribution='gaussian', mean=0., stddev=100
# Sumamos el ruido a la imagen
image += noise
# Muestra la imagen con ruido
plt.figure()
plt.imshow(image, origin='lower', cmap='gray')
plt.show()

# Apartado 4
from astropy.convolution import convolve
from astropy.convolution import Gaussian2DKernel

# Definimos el kernel
sigmaG = 4.0
gauss = Gaussian2DKernel(sigmaG)
# Muestra el kernel
plt.imshow(gauss.array, cmap='gray')
plt.show()
image_conv = convolve(image, gauss)
# Muestra la imagen convolucionada
plt.figure()
plt.imshow(image_conv, cmap='gray')
plt.show()
threshold = 5*(image_conv.std()) # Umbral de detección

# Apartado 5
# Busca los picos en la imagen
peaks_tbl = find_peaks(image_conv, threshold)

# Muestra la tabla con los picos
print(peaks_tbl)

tabla = QTable()
tabla['amplitude'] = peaks_tbl['peak_value']
tabla['x_mean'] = peaks_tbl['x_peak']
tabla['y_mean'] = peaks_tbl['y_peak']

imagen = make_gaussian_sources_image(shape, tabla)

# Apartado 6
plt.figure()
plt.imshow(imagen, origin='lower', cmap='gray')
plt.scatter(tabla['x_mean'], tabla['y_mean'], color='red', s=10)
plt.show()
plt.figure()
plt.subplot(1,2,1)
plt.imshow(image, cmap='gray')
plt.title("Fuentes originales")
plt.subplot(1,2,2)
plt.imshow(imagen, cmap='gray')
plt.scatter(tabla['x_mean'], tabla['y_mean'], color='red', s=10)
plt.title("Picos detectados")

plt.show()
```

Loading [MathJax]/extensions/Safe.js

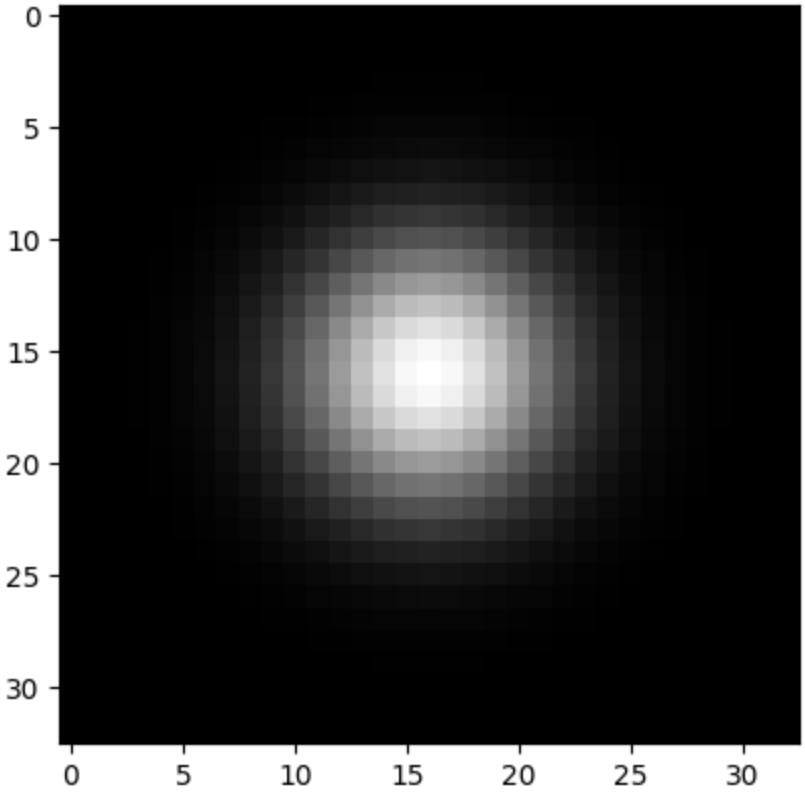
```
    amplitude           x_mean       ...      y_stddev            theta
---------------- ----------------- ... ---------------- ----------------
--
873.6030612791037 208.93457900829299 ... 4.270378852875055  0.31298193272827
52
701.0042514174953 54.406139123115096 ... 4.650293770030598  1.38553174339677
84
975.1658950152662 411.75004814114885 ... 4.154411640166333  0.93664140245886
56
  725.35582839259 345.51700482777017 ... 4.117560848857455  1.71760789570533
76
977.4401653078917  419.1237151665717 ... 4.998370725281204  1.01527606210852
17
675.8152945795517 224.11176375856718 ... 4.651131270264389   2.0792341650562
07
911.9556879157414 151.17136401439478 ... 4.145575948005618  1.59789562756002
07
787.7768897556691 132.49309913467826 ... 4.628563880492301  2.12450277085503
06
736.1810044691401 104.20464659188644 ... 4.477599131485244  0.80878535490539
25
 867.057572813751 239.59207915545454 ... 4.269150153046253   1.3600904901621
71
 879.056707932486  397.4241189274079 ...   4.4794414620855  1.56755632165044
11
748.7634383870288  292.6935692734204 ... 4.139170577986452   0.7522815815354
25
 806.401825691837 159.03610980423437 ... 4.056558652342827 0.187754798549035
82
853.1227096803419  49.12706173977632 ... 4.529898916007582   2.4454594090687
67
889.4164719037001  98.96286646622637 ... 4.002425042307521   1.3920658701391
53
```

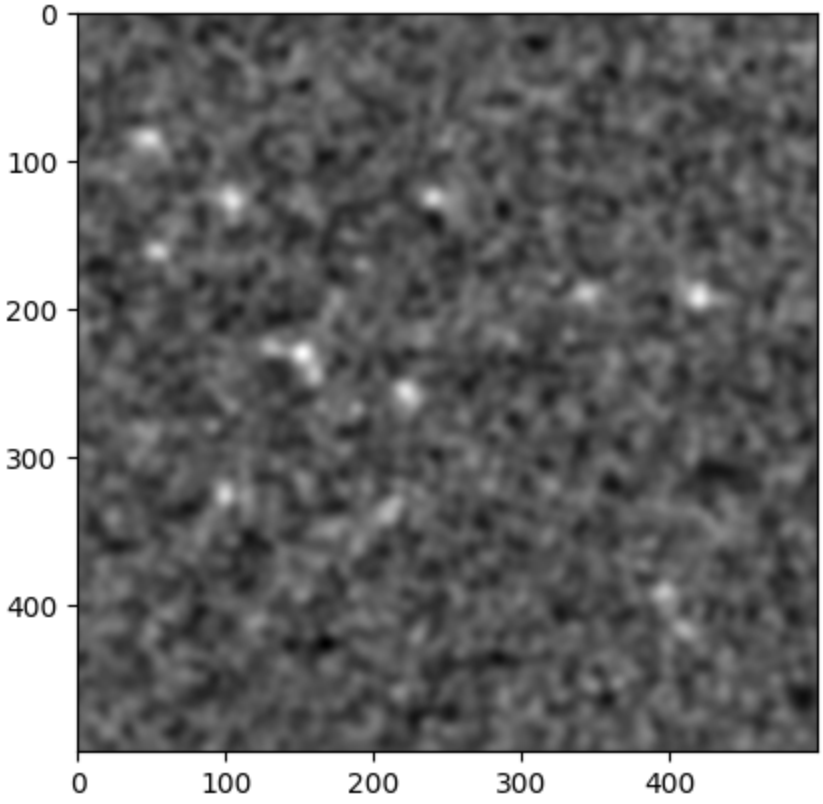
Loading [MathJax]/extensions/Safe.js





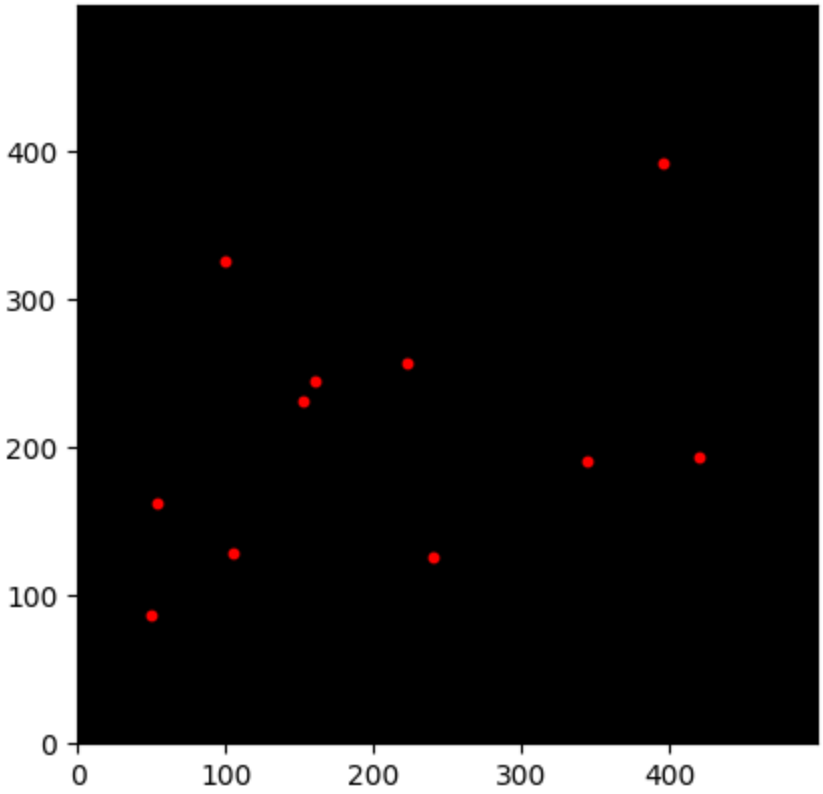
Loading [MathJax]/extensions/Safe.js

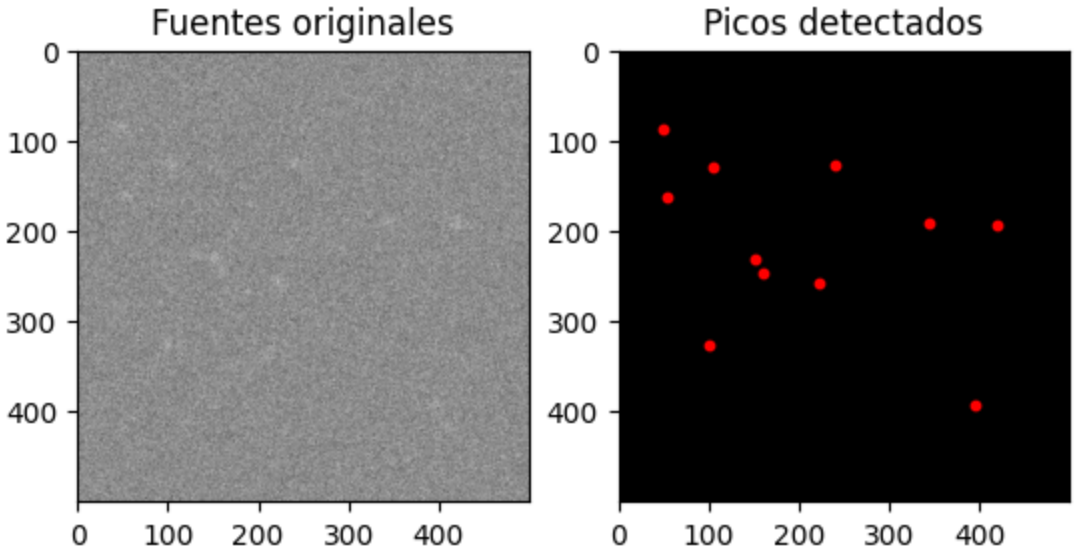




Loading [MathJax]/extensions/Safe.js

x_peak	y_peak	peak_value
49	86	529.915117357819
240	126	498.52384069595314
105	128	546.2647436643848
54	162	470.2975609622761
344	190	443.67198523374753
420	193	585.9958764364236
152	231	637.3640812232587
160	245	413.5599654367508
222	257	541.1340427513858
100	326	493.9053197689546
396	392	418.70419733726834





Loading [MathJax]/extensions/Safe.js

In [ ]:

```
# PROBLEMA 7
# Apartado 1
from astropy import constants as const
import astropy.units as u
import math
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
#Charge the images h m51 h s20 drz sci.fits, h m51 v s20 drz sci.fits and h
#We will use the function fits.open to open the images.
from astropy.io import fits
import numpy as np
imagen_h = fits.open('h_m51_h_s20_drz_sci.fits')
imagen_v = fits.open('h_m51_v_s20_drz_sci.fits')
imagen_b = fits.open('h_m51_b_s20_drz_sci.fits')
#Which are the dimensions of the images?
imagen_h.info()
imagen_v.info()
imagen_b.info()

# Apartado 2
#Visualize the images in a figure with 3 subplots.
plt.figure()
plt.subplot(1,3,1)
plt.imshow(imagen_h[0].data, cmap='gray')
plt.title('Imagen h')
plt.subplot(1,3,2)
plt.imshow(imagen_v[0].data, cmap='gray')
plt.title('Imagen v')
plt.subplot(1,3,3)
plt.imshow(imagen_b[0].data, cmap='gray')
plt.title('Imagen b')
plt.show()

# Apartado 3
image_vector_h = imagen_h[0].data.flatten()
image_vector_v = imagen_v[0].data.flatten()
image_vector_b = imagen_b[0].data.flatten()

# Apartado 4
plt.figure(figsize=(12, 4))
# Histograma de la imagen h
plt.subplot(1, 3, 1)
plt.hist(image_vector_h, bins=100, log=True)
plt.title('Histograma de la imagen h')
plt.xlabel('Valor de píxel')
plt.ylabel('Frecuencia')
# Histograma de la imagen v
plt.subplot(1, 3, 2)
plt.hist(image_vector_v, bins=100, log=True)
plt.title('Histograma de la imagen v')
plt.xlabel('Valor de píxel')
plt.ylabel('Frecuencia')
# Histograma de la imagen b
plt.subplot(1, 3, 3)
```

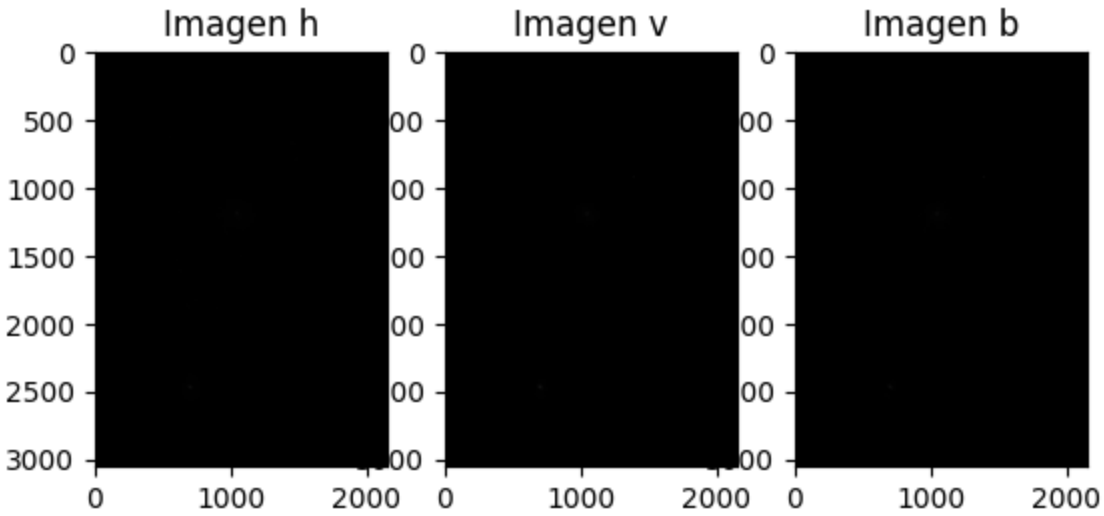
Loading [MathJax]/extensions/Safe.js

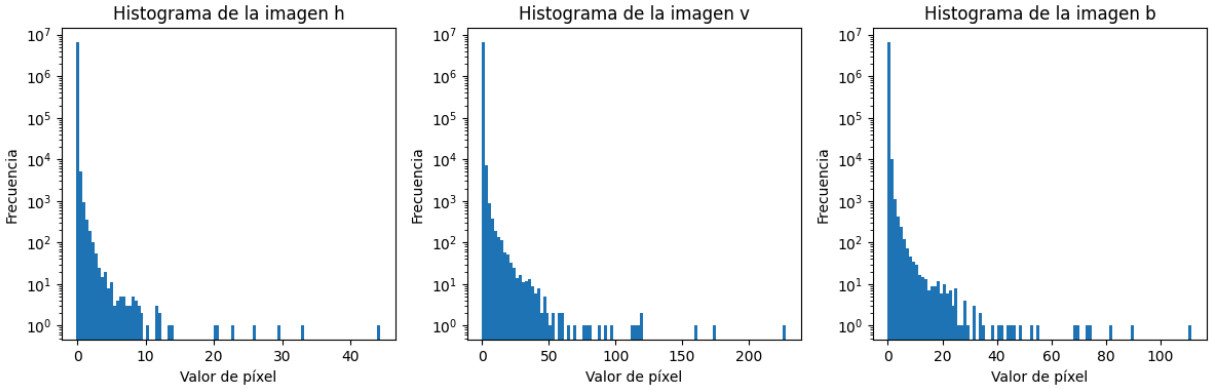
```
plt.hist(image_vector_b, bins=100, log=True)
plt.title('Histograma de la imagen b')
plt.xlabel('Valor de píxel')
plt.ylabel('Frecuencia')

plt.tight_layout()
plt.show()

# Apartado 5
percentil_h = np.percentile(image_vector_h, 99)
percentil_v = np.percentile(image_vector_v, 99)
percentil_b = np.percentile(image_vector_b, 99)

print("Percentil del 99% de la imagen h:", percentil_h)
print("Percentil del 99% de la imagen v:", percentil_v)
print("Percentil del 99% de la imagen b:", percentil_b)

# Apartado 6
plt.figure()
plt.subplot(1,3,1)
plt.imshow(imagen_h[0].data, cmap='gray', clim=(0, percentil_h))
plt.title('Imagen h')
plt.subplot(1,3,2)
plt.imshow(imagen_v[0].data, cmap='gray', clim=(0, percentil_v))
plt.title('Imagen v')
plt.subplot(1,3,3)
plt.imshow(imagen_b[0].data, cmap='gray', clim=(0, percentil_b))
plt.title('Imagen b')
plt.show()

# Apartado 7
# Normalizar las imágenes
imagen_h_norm = imagen_h[0].data / percentil_h
imagen_v_norm = imagen_v[0].data / percentil_v
imagen_b_norm = imagen_b[0].data / percentil_b
# Ajustar los valores al rango [0, 1]
imagen_h_norm = np.clip(imagen_h_norm, 0, 1)
imagen_v_norm = np.clip(imagen_v_norm, 0, 1)
imagen_b_norm = np.clip(imagen_b_norm, 0, 1)

# Apartado 8
# Define the dimensions of the array
height = 3050
width = 2150
# Create the array with zeros
rgb = np.zeros((height, width, 3))
# Assign normalized images to each column
rgb[:, :, 0] = imagen_h_norm
rgb[:, :, 1] = imagen_v_norm
rgb[:, :, 2] = imagen_b_norm

# Apartado 9
import matplotlib.pyplot as plt

plt.imshow(rgb)
plt.title('Reconstructed Color Image')
```

Loading [MathJax]/extensions/Safe.js

```
plt.axis('off')
plt.show()
```

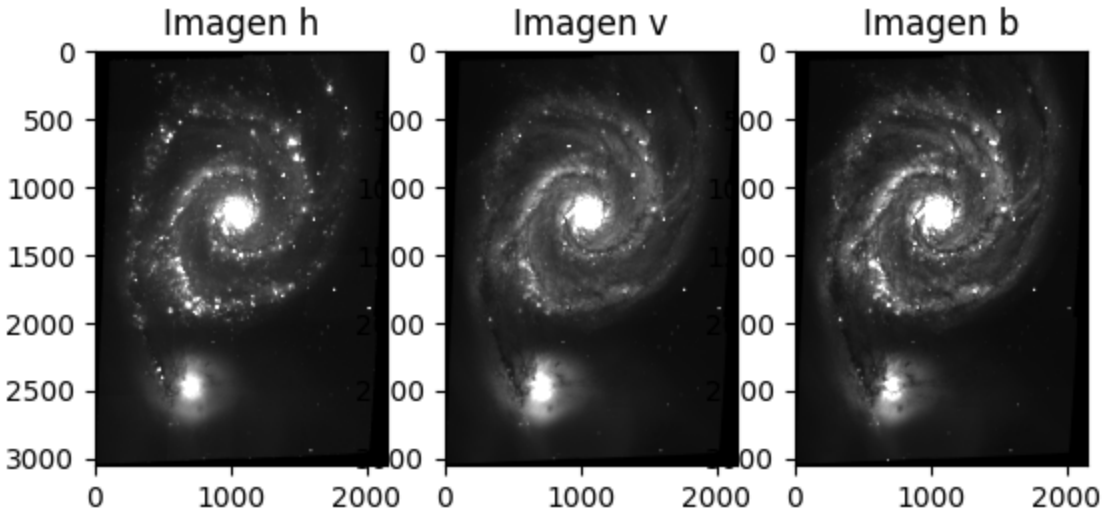
```
Filename: h_m51_h_s20_drz_sci.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU    1691   (2150, 3050)   float32
Filename: h_m51_v_s20_drz_sci.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU    1763   (2150, 3050)   float32
Filename: h_m51_b_s20_drz_sci.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU    1763   (2150, 3050)   float32
Dimensiones de la imagen h: (3050, 2150)
```

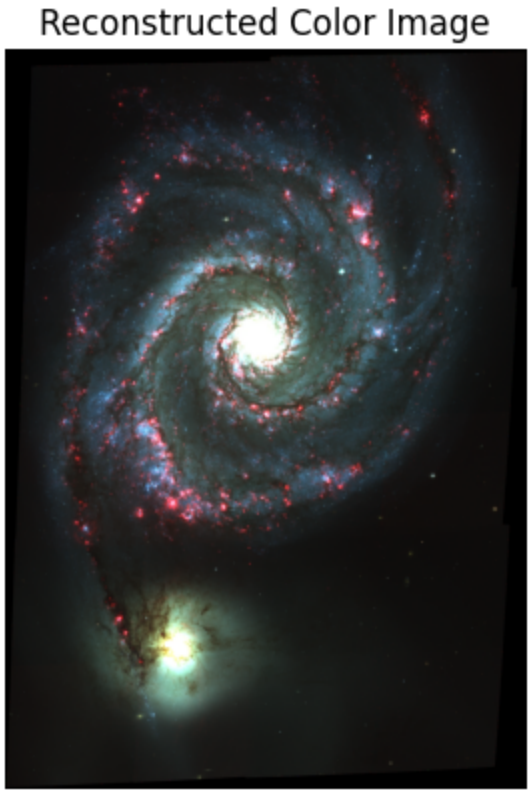





```
Percentil del 99% de la imagen h: 0.10837415121495718
Percentil del 99% de la imagen v: 0.7322072076797466
Percentil del 99% de la imagen b: 0.42744414389133223
```

Loading [MathJax]/extensions/Safe.js





Loading [MathJax]/extensions/Safe.js