Laboratorium 2

Autorzy: Krzysztof Zalewa 273032, Michał Pakuła 272828
Data: 7 Kwietnia 2025

Ćwiczenie 5

Napisz skrypt w Pythonie/Matlabie umożliwiający wczytywanie i wizualizację badanych obrazów. Program powinien umożliwiać:

1. wyświetlanie obrazu wczytanego z pliku o podanej nazwie,
2. sporządzenie wykresów zmian poziomu szarości wzdłuż wybranej linii poziomej lub pionowej o zadanej współrzędnej,
3. wybór podobrazu (prostokątnego obszaru) o podanych współrzędnych oraz jego zapis do pliku o zadanej nazwie.

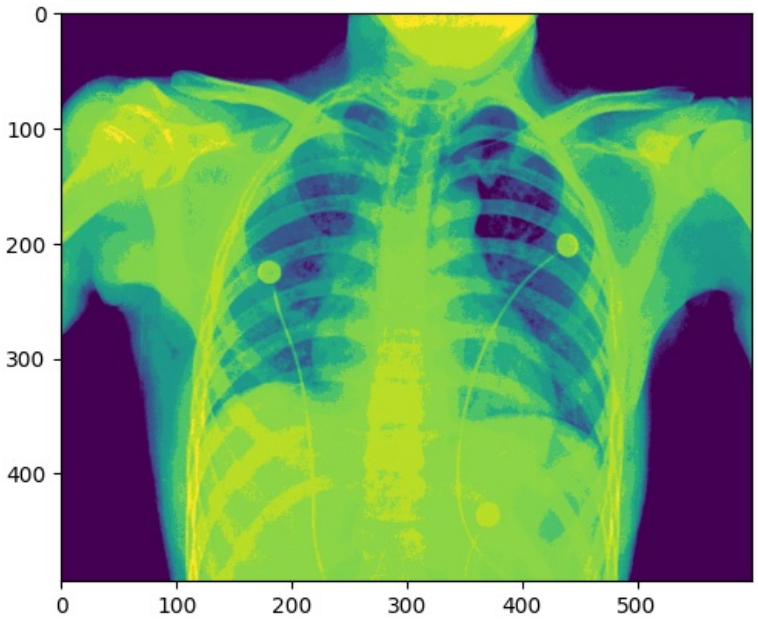
In [1]:
```
import matplotlib.pyplot as plt
import tifffile as tiff
import numpy as np
import os
```

Zadanie 1

In [2]:
```
# Załadowanie pliku .tiff
if os.name == 'nt':
    file_name = "./src/"+input("Podaj nazwe pliku z danymi: ")
elif os.name == 'posix':
    file_name = ".//src//"+input("Podaj nazwe pliku z danymi: ")
else:
    print("Nieznany system")
img = tiff.imread(file_name)
```

In [3]:
```
#Wyświetlenie załadowanego obrazu
plt.figure()
plt.imshow(img)

plt.show()
```



Zadanie 2

Stworzenie histogramu obrazu

Histogram jest tworzony na podstawie jednej linii (poziomej lub pionowej) która jest wybrana przez użytkownika

In [ ]:
```
if len(img.shape)==3:
    img = np.mean(img,axis=2).astype(np.uint8)
mode = input("Podaj :")

if( mode == "pozioma"):
    line_num = int(input("Podaj: "))
    gray_val = img[line_num,:]

elif(mode == "pionowa"):
```

```
    line_num = int(input("Podaj: "))
    gray_val = img[:,line_num]

else:
    print("Eee")

plt.figure()
plt.plot(gray_val)
plt.title('Histogram skali szarości')
plt.xlabel('Wartość pikseli')
plt.ylabel('Częstość')
plt.show()
```

Zadanie 3

Stworzenie wycinka obrazu

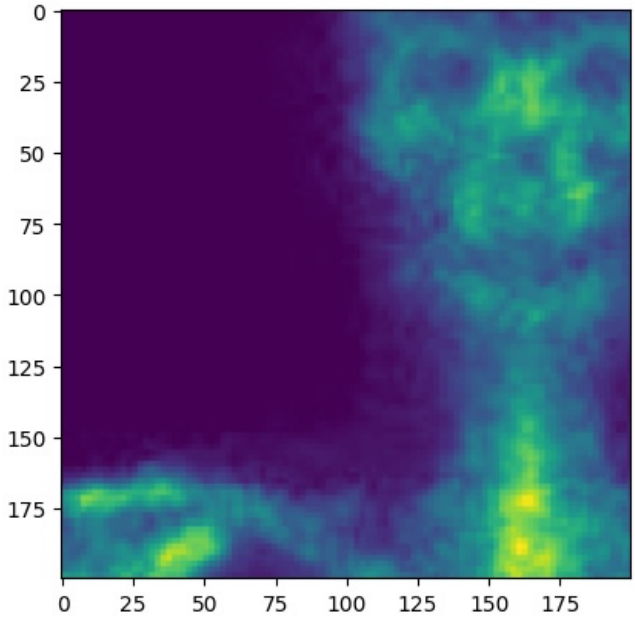
Użytkownik podaje współrzędne a następnie wyznacza szrokość oraz wysokość wycinka od danego punktu

In [12]:
```
try:
    x, y, w, h = map(int, input("Podaj współrzędne X, Y, szerokość i wysokość prostokąta (oddzielone spacjami):

    if w <= 0 or h <= 0:
        print("Szerokość i wysokość muszą być dodatnie.")
except ValueError:
        print("To nie są liczby całkowite. Proszę spróbować ponownie.")
region_of_intrest = img[y:y+h , x:x+w]

plt.figure()
plt.imshow(region_of_intrest)

plt.show()
```


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

Ćwiczenie 6

Zaobserwuj działanie następujących przekształceń punktowych:

1. Mnożenie obrazu przez stałą
2. Transformacja logarytmiczna
3. Zmiana dynamiki skali szarości (kontrastu)
4. Korekcja gamma

In [35]:
```
import matplotlib.pyplot as plt
import tifffile as tiff
import numpy as np
import os
```

Załadowanie wyznaczonych plików:

In [36]:
```
# Załadowanie pliku .tiff
img_a = tiff.imread("src/pollen-dark.tif")
img_b = tiff.imread("src/spectrum.tif")
img_c  = tiff.imread("src/einstein-low-contrast.tif")
img_d = tiff.imread("src/aerial_view.tif")
```

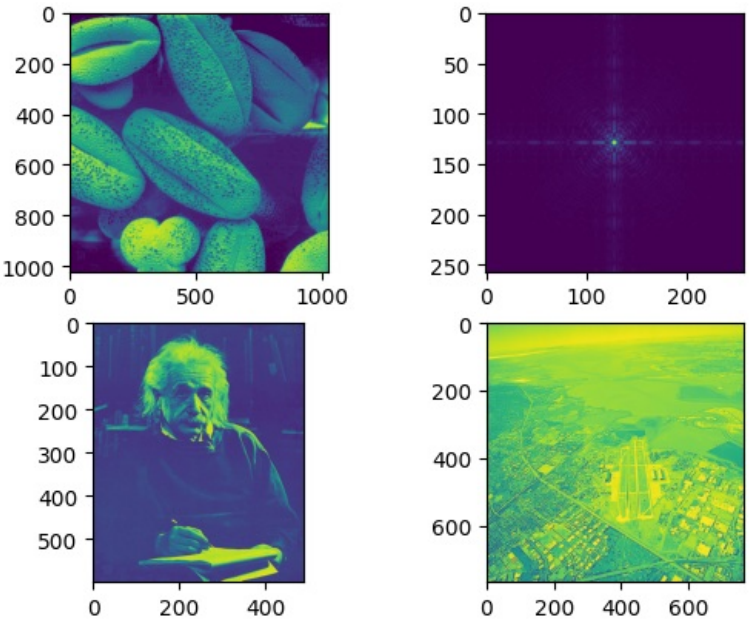
Obrazy bazowe:

In [37]:
```
#Wyświetlenie załadowanego obrazu
plt.figure()
plt.subplot(2,2,1)
plt.imshow(img_a)

plt.subplot(2,2,2)
plt.imshow(img_b)

plt.subplot(2,2,3)
plt.imshow(img_c)

plt.subplot(2,2,4)
plt.imshow(img_d)
plt.show()
```



Funkcje przekształcające

Przyjmujemy że 'r', to nasz przetwarzany obraz

c jest stałą podaną przez użytkownika

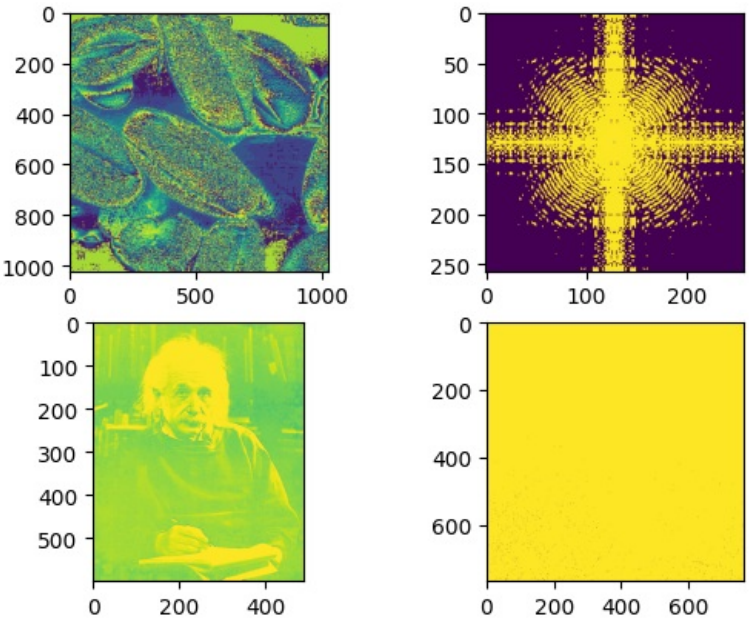
1. T(r) = c*r
2. T(r) = c*log(1+r)
3. T(r) = 1/(1+(m/r)^e)

m oraz *e* są ustalonymi wartościami całkowitymi

4. T(r) = c*(r^gamma); gdzie c>0 oraz gamma>0

In [41]:
```
m = 0.8
e = 20
gamma = 0.2
plt.figure()
#pollen-dark
plt.subplot(2,2,1)
c = int(input("Podaj"))
img_a = img_a*c
plt.imshow(img_a)
#spectrum
plt.subplot(2,2,2)
img_b = c * np.log(1+img_b)
plt.imshow(img_b)
#einstein-low-contrast
plt.subplot(2,2,3)
img_c = 1/(1+(m/img_c)**e)
plt.imshow(img_c)
#aerial-view
plt.subplot(2,2,4)
if(gamma>0 and c>0):
    img_d = c * pow(img_d,gamma)
    plt.imshow(img_d)

plt.show()
```



Ćwiczenie 7

Wypróbuj działanie wyrównywania histogramu na przykładowych obrazach. By zaobserwować skuteczność procedury, poddaj wyrównywaniu obrazy zbyt ciemne i zbyt jasne.

Narysować histogramy obrazów przed i po wyrównaniu.

In [1]:
```
import matplotlib.pyplot as plt
import tifffile as tiff
from skimage import exposure
import numpy as np
```

In [9]:
```
# Załadowanie pliku .tiff
img_a = tiff.imread("src/chest-xray.tif")
img_b = tiff.imread("src/pollen-dark.tif")
img_c  = tiff.imread("src/pollen-ligt.tif")
img_d = tiff.imread("src/pollen-lowcontrast.tif")
img_e = tiff.imread("src/pout.tif")
img_f = tiff.imread("src/spectrum.tif")
```

In [13]:
```
def histOfImg(img,title,index):
    if len(img.shape)==3:
        vals = np.mean(img,axis=2).astype(np.uint8)
    else:
        vals = img

    counts, bins = np.histogram(vals, range(257))

    plt.subplot(2,2,index)
    plt.imshow(img)
    plt.title(title)

    plt.subplot(2,2,index+1)
    plt.bar(bins[:-1] - 0.5, counts, width=1, edgecolor='none')
    plt.xlim([-0.5, 255.5])
    plt.title("Histogram ")
```

In [17]:
```
def hist(img):
    plt.figure(figsize=(10, 8))
    histOfImg(img,"Obraz orginalny",1)
    equ = exposure.equalize_hist(img)
    equ_uint8 = (equ * 255).astype(np.uint8)
    histOfImg(equ_uint8,"Obraz wyrównany",3)
    plt.tight_layout()
    plt.show()
```

In [18]:
```
print("Image: chest-xray.tif")
hist(img_a)

print("Image: pollen-dark.tif")
hist(img_b)

print("Image: pollen-ligt.tif")
hist(img_c)

print("Image: pollen-lowcontrast.tif")
hist(img_d)

print("Image: chest-pout.tif")
hist(img_e)

print("Image: chest-spectrum.tif")
hist(img_f)
```

```
Image: chest-xray.tif
```

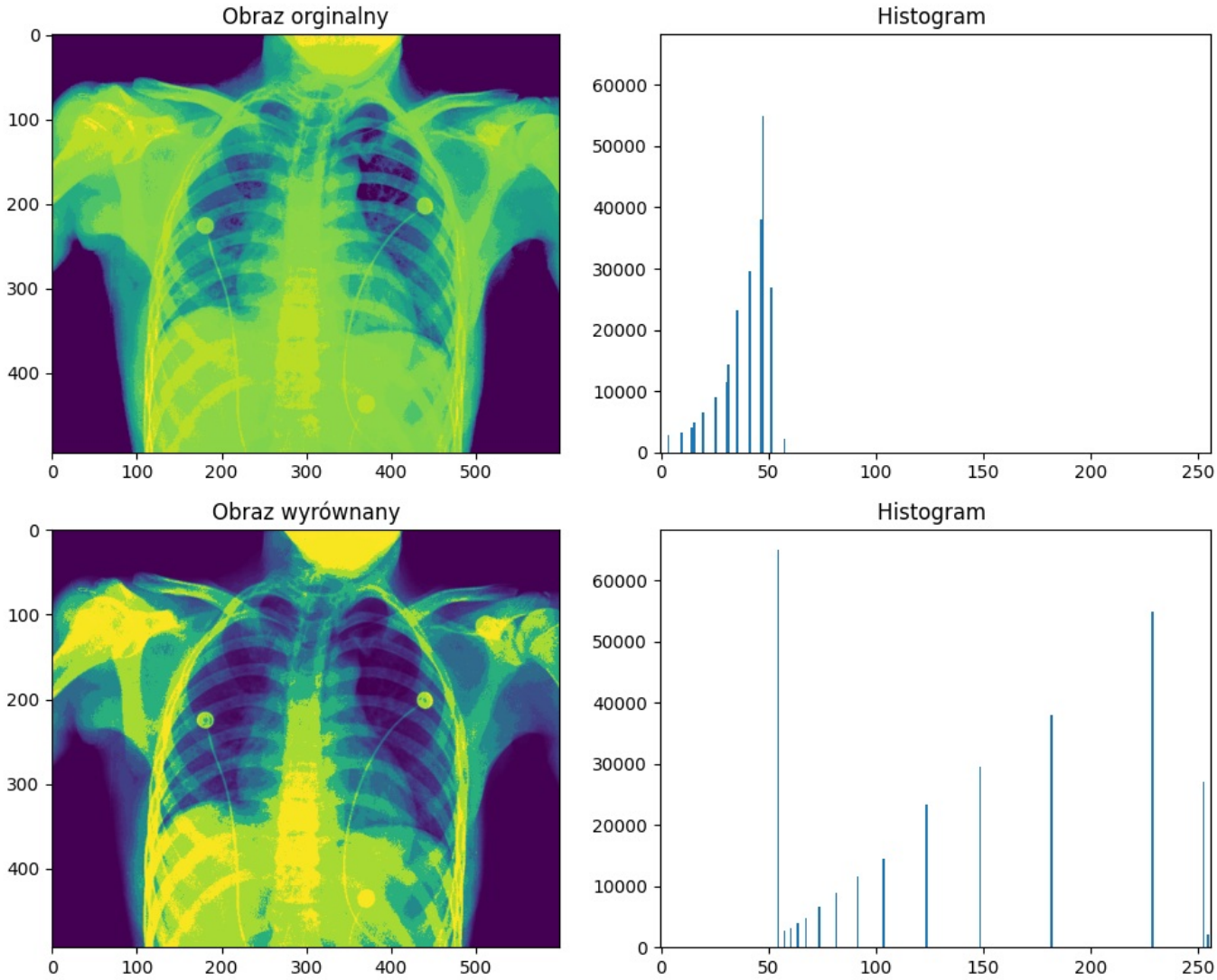
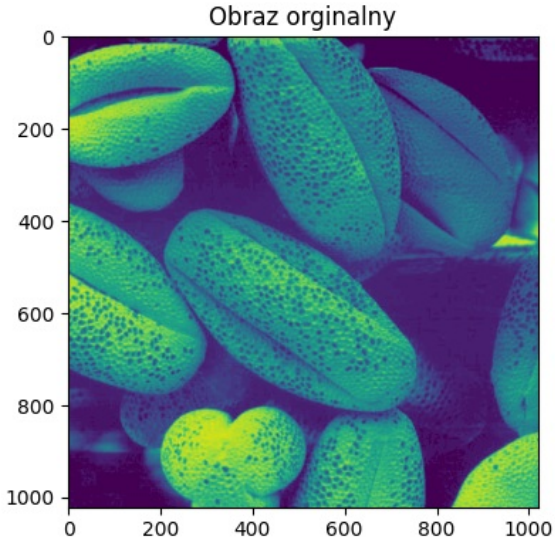
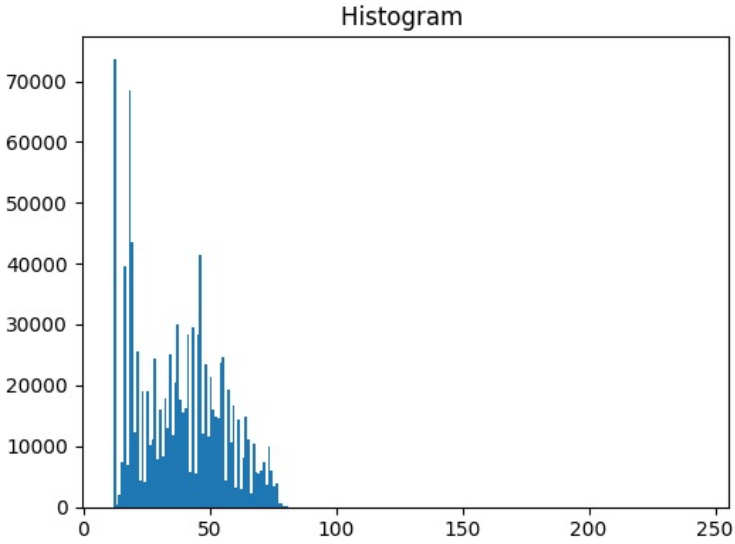
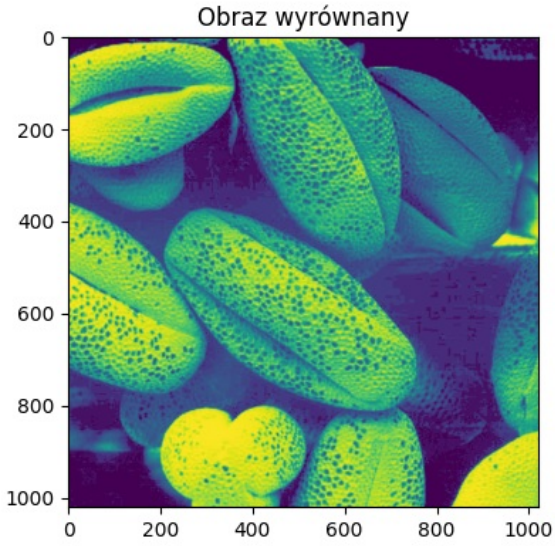


Image: pollen-dark.tif







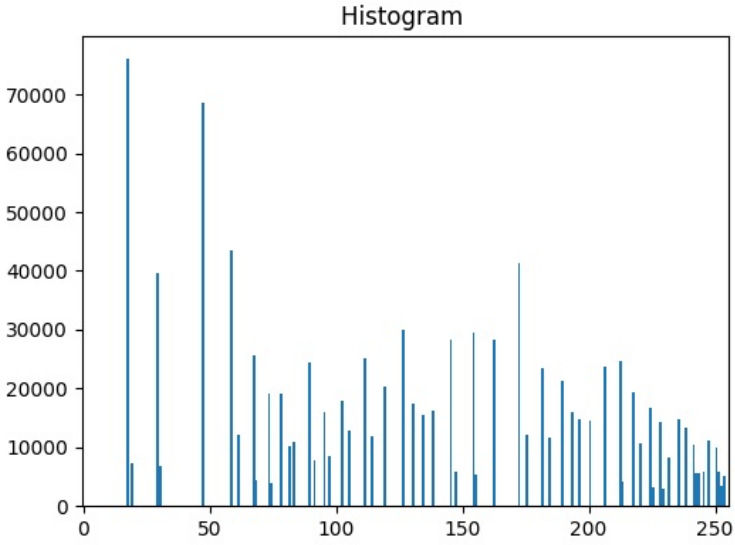
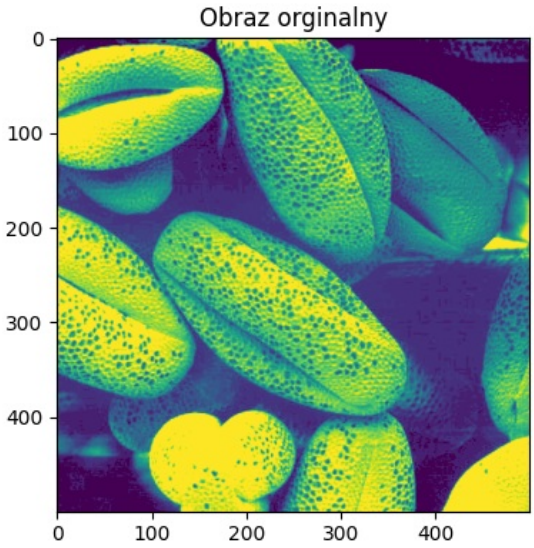
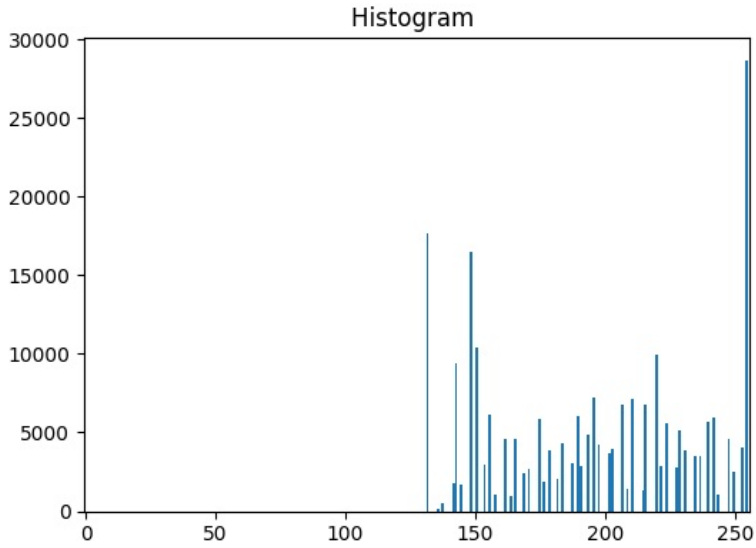
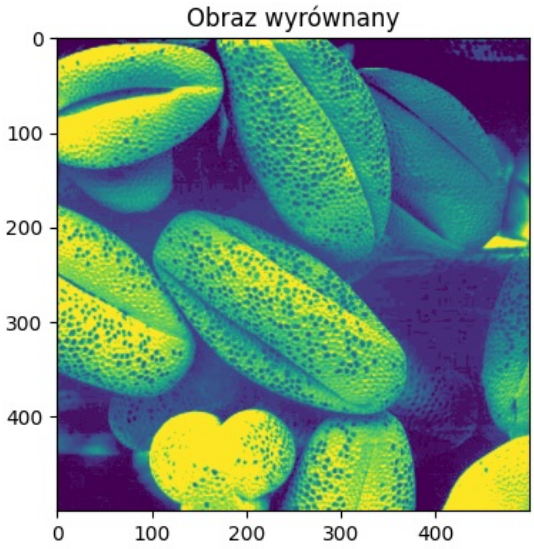


Image: pollen-ligt.tif







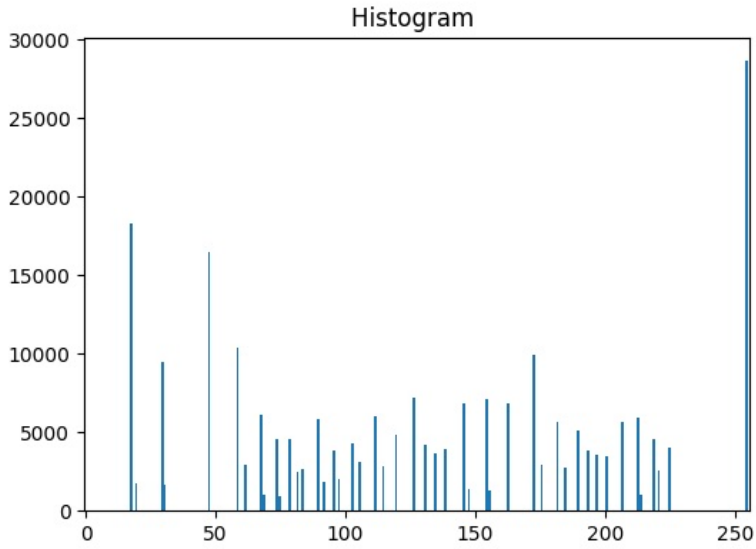
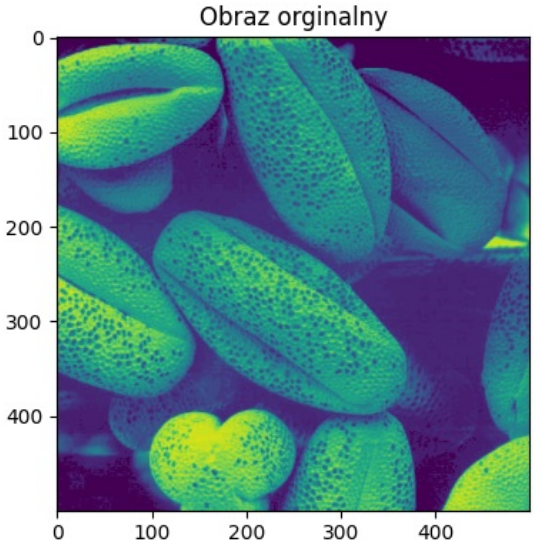
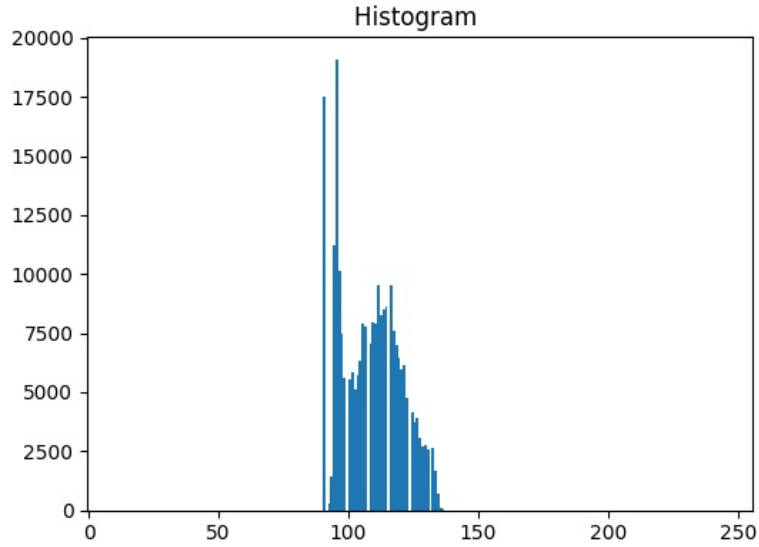
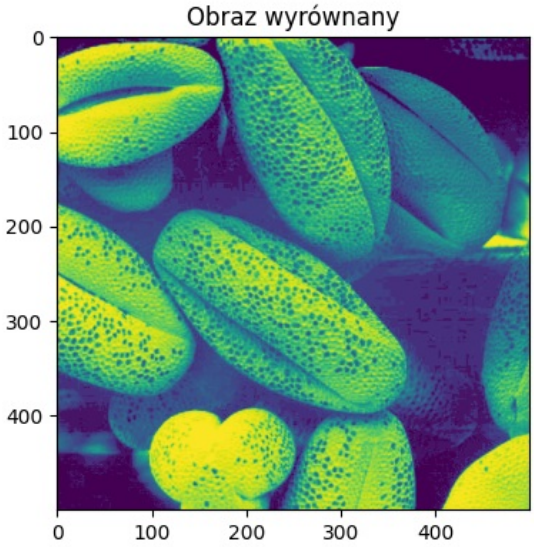


Image: pollen-lowcontrast.tif







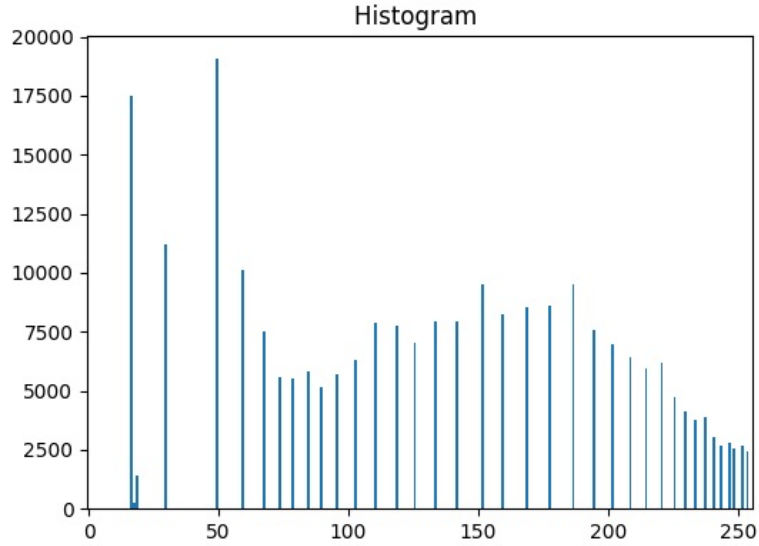


Image: chest-pout.tif

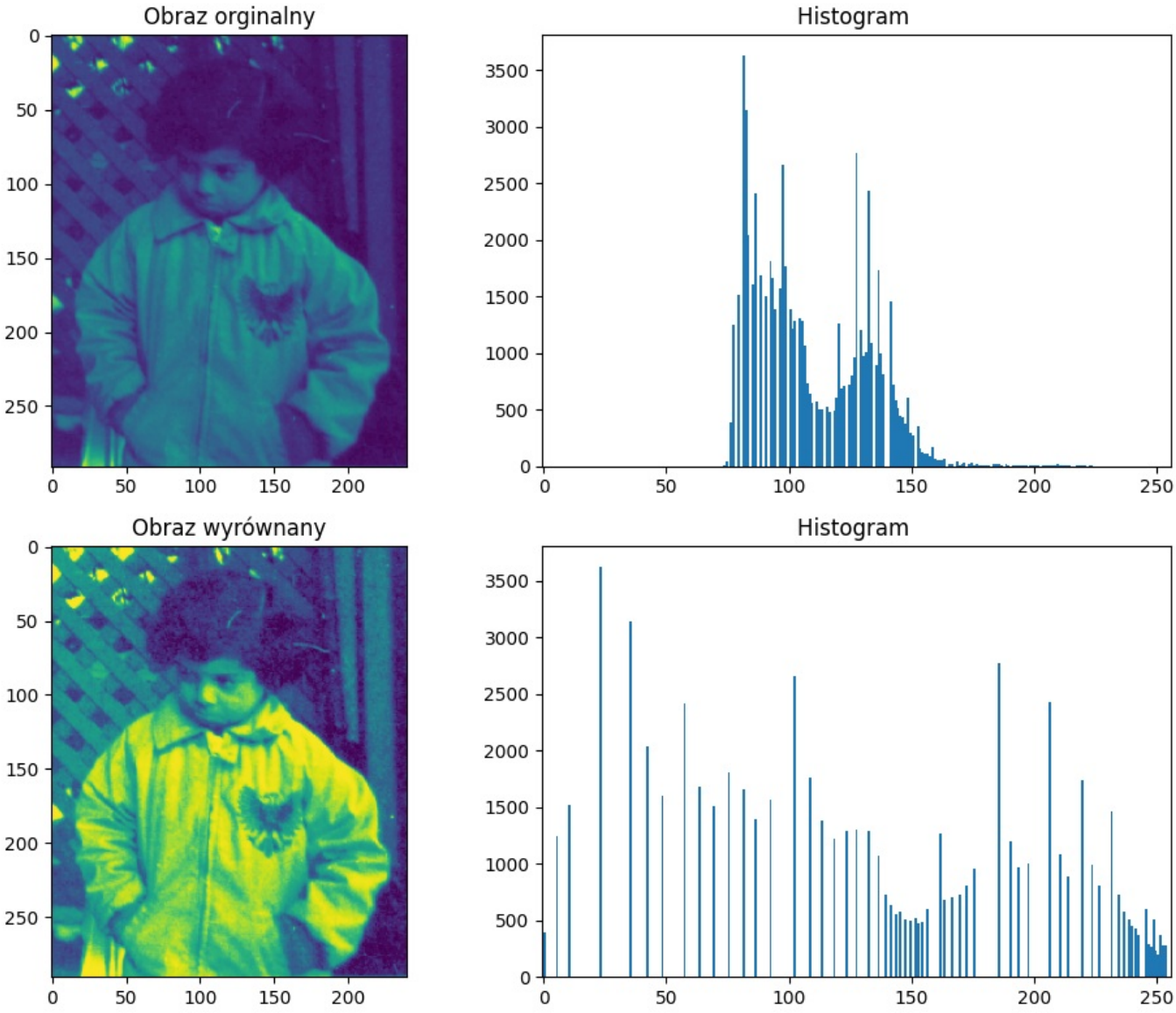
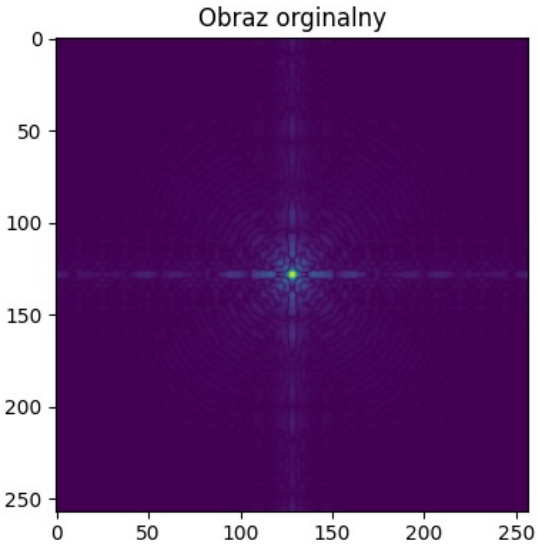
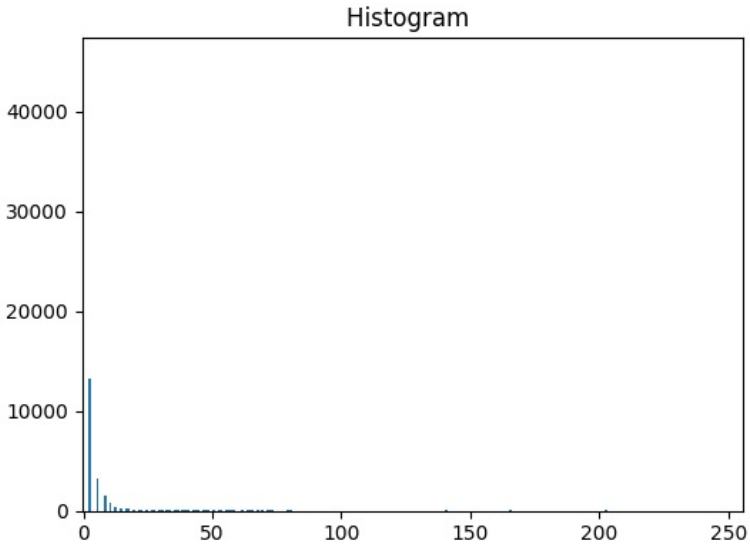
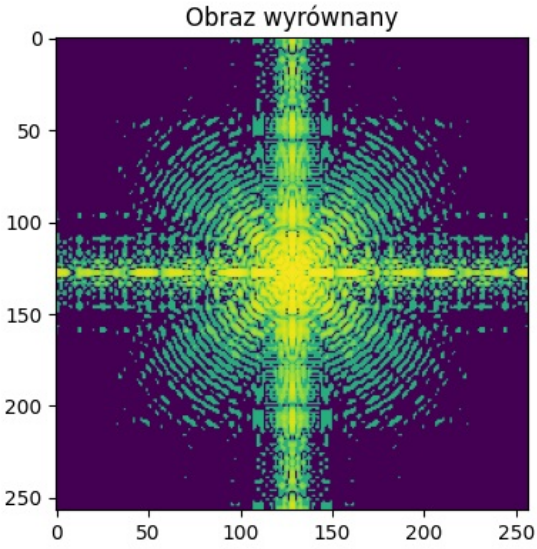
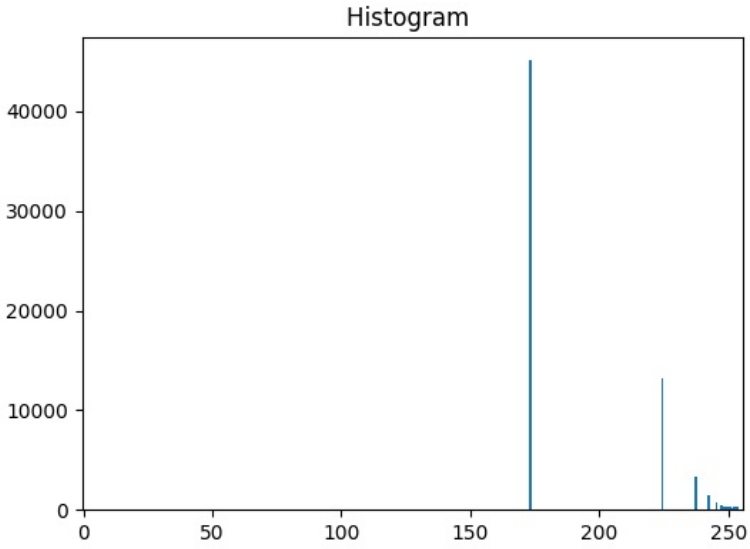


Image: chest-spectrum.tif


Obraz orginalny
0
50
100
150
200
250
0
50
100
150
200
250


Histogram
40000
30000
20000
10000
0
0
50
100
150
200
250


Obraz wyrównany
0
50
100
150
200
250
0
50
100
150
200
250


Histogram
40000
30000
20000
10000
0
0
50
100
150
200
250

In [14]:
```
import matplotlib.pyplot as plt
import tifffile as tiff
from skimage import exposure
import skimage.morphology as morp
from skimage.filters import rank
```

In [3]:
```
# Załadowanie pliku .tiff
img = tiff.imread("src/hidden-symbols.tif")
```

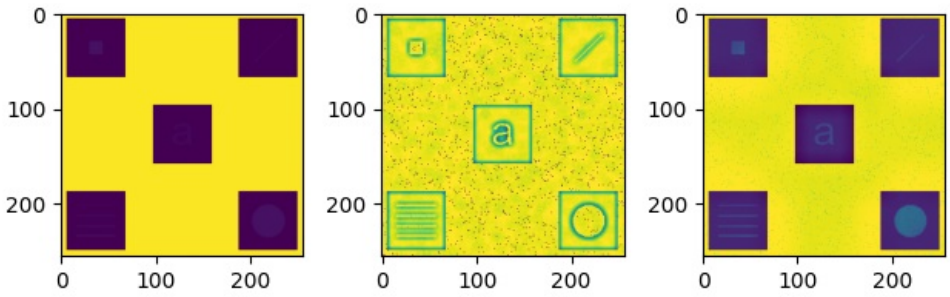
In [4]:
```
#Wyświetlenie załadowanego obrazu
img_local = rank.equalize(img, morp.disk(5))
img_stat = exposure.equalize_adapthist(img, clip_limit=0.15)

plt.figure()
plt.subplot(1,3,1)
plt.imshow(img)

plt.subplot(1,3,2)
plt.imshow(img_local)

plt.subplot(1,3,3)
plt.imshow(img_stat)

plt.tight_layout()
plt.show()
```



Ćwiczenie 9

Zbadaj skuteczność redukcji szumu typu „sól i pieprz" za pomocą

1. liniowego filtra uśredniającego z kwadratową maską, rozpoczynając od maski rozmiaru 3 × 3.
2. nieliniowego filtra medianowego
3. filtrów minimum i maksimum.

In [1]:
```
import matplotlib.pyplot as plt
import tifffile as tiff
import skimage.morphology as morph
from skimage.filters import rank
```

In [2]:
```
# Załadowanie pliku .tiff
img_a = tiff.imread("src/cboard_pepper_only.tif")
img_b = tiff.imread("src/cboard_salt_only.tif")
img_c = tiff.imread("src/cboard_salt_pepper.tif")
```

Zadanie 1

Filtr uśredniający z kwadratową maską 3x3

In [3]:
```
def meanFilter(img):
    plt.subplot(2,3,2)
    mean_img = rank.mean(img,morph.footprint_rectangle((3,3)))
    plt.imshow(mean_img)
    plt.title("Filtr uśredniający")
```

Zadanie 2

Nieliniowy filtr medianowy

In [4]:
```
def mediFilter(img):
    plt.subplot(2,3,3)
    medi_img = rank.median(img,morph.footprint_rectangle((3,3)))
    plt.imshow(medi_img)
    plt.title("Filtr medianowy")
```

Zadanie 3

Filtry minimum i maximum

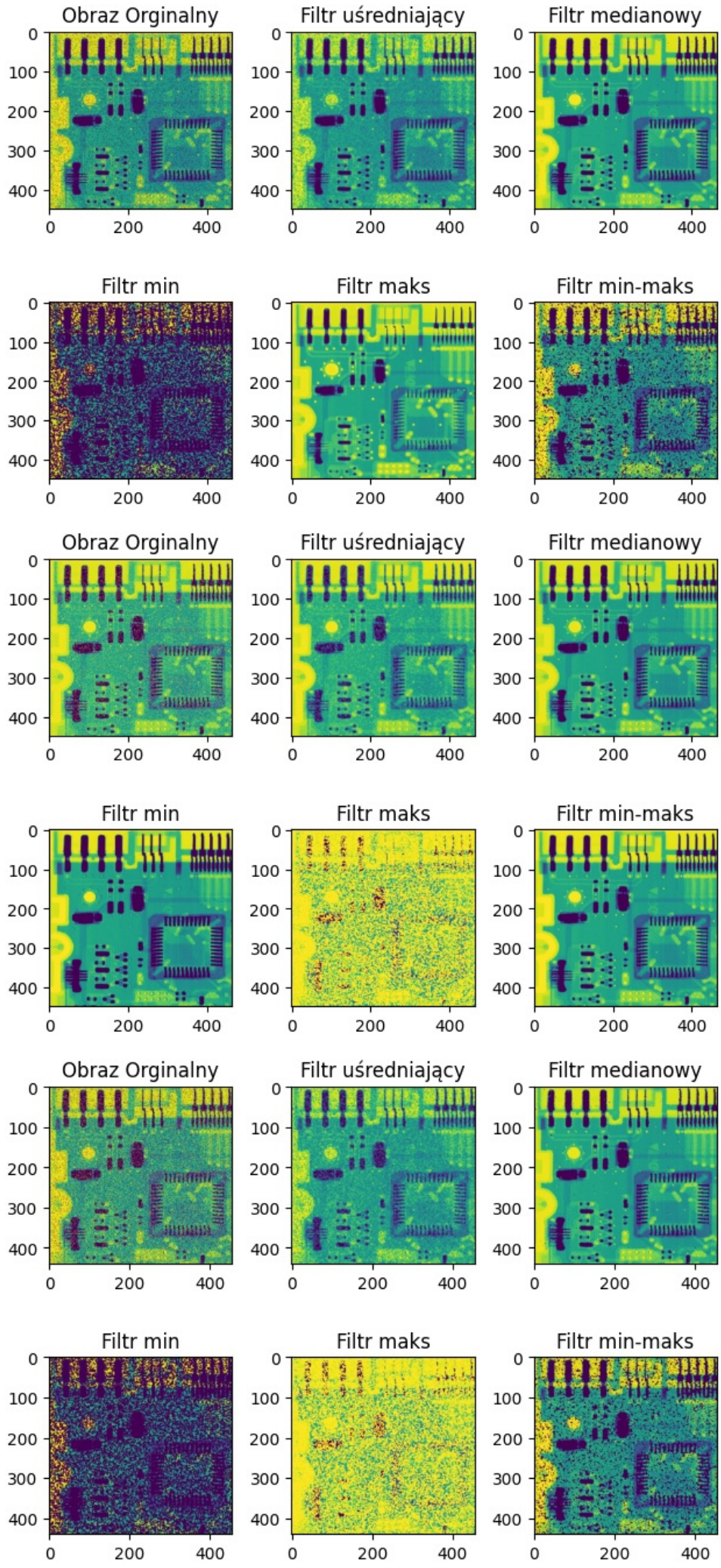
In [ ]:
```
def minMaxFilter(img):
    plt.subplot(2,3,4)
    min_img = rank.minimum(img,morph.footprint_rectangle((3,3)))
    plt.imshow(min_img)
    plt.title("Filtr min")

    plt.subplot(2,3,5)
    max_img = rank.maximum(img,morph.footprint_rectangle((3,3)))
    plt.imshow(max_img)
    plt.title("Filtr maks")

    plt.subplot(2,3,6)
    minMax_img = rank.maximum(min_img,morph.footprint_rectangle((3,3)))
    plt.imshow(minMax_img)
    plt.title("Filtr min-maks")
```

In [6]:
```
def display(img):
    plt.subplot(2,3,1)
    plt.imshow(img)
    plt.title("Obraz Orginalny")
    meanFilter(img)
    mediFilter(img)
    minMaxFilter(img)
    plt.tight_layout()
    plt.show()
```

In [7]:
```
display(img_a)
display(img_b)
display(img_c)
```


Obraz Orginalny
Filtr uśredniający
Filtr medianowy
Filtr min
Filtr maks
Filtr min-maks
Obraz Orginalny
Filtr uśredniający
Filtr medianowy
Filtr min
Filtr maks
Filtr min-maks
Obraz Orginalny
Filtr uśredniający
Filtr medianowy
Filtr min
Filtr maks
Filtr min-maks

Zadanie 10

Zbadaj działanie dolnoprzepustowych filtrów uśredniającego i gaussowskiego dla danych obrazów. Zaobserwuj wpływ rozmiaru masek na wynik filtracji.

In [1]:
```
import matplotlib.pyplot as plt
import tifffile as tiff
import skimage.filters as flt
import skimage.morphology as morph
from skimage.filters import rank
```

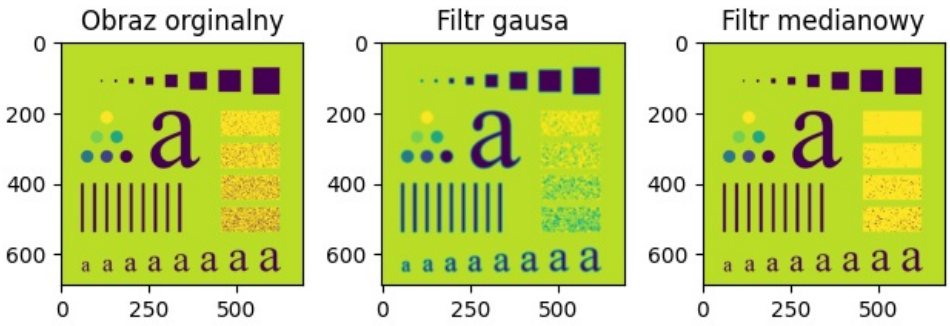
In [2]:
```
# Załadowanie pliku .tiff
img_a = tiff.imread("src/characters_test_pattern.tif")
img_b = tiff.imread("src/zoneplate.tif")
```

In [3]:
```
plt.subplot(1,3,1)
plt.imshow(img_a)
plt.title("Obraz orginalny")

plt.subplot(1,3,2)
gaus_a = flt.gaussian(img_a,sigma=2)
plt.imshow(gaus_a)
plt.title("Filtr gausa")

plt.subplot(1,3,3)
medi_img = rank.median(img_a,morph.footprint_rectangle((3,3)))
plt.imshow(medi_img)
plt.title("Filtr medianowy")

plt.tight_layout()
plt.show()
```

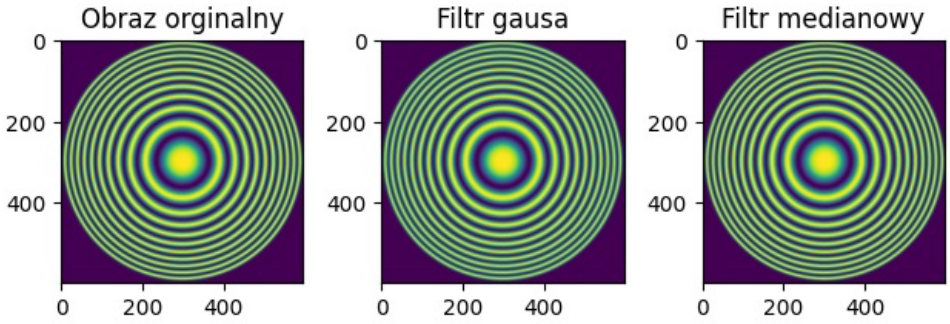


In [4]:
```
plt.subplot(1,3,1)
plt.imshow(img_b)
plt.title("Obraz orginalny")

plt.subplot(1,3,2)
gaus_b = flt.gaussian(img_b,sigma=1)
plt.imshow(gaus_b)
plt.title("Filtr gausa")

plt.subplot(1,3,3)
medi_img = rank.median(img_b,morph.footprint_rectangle((3,3)))
plt.imshow(medi_img)
plt.title("Filtr medianowy")

plt.tight_layout()
plt.show()
```



Ćwiczenie 11

Wykrywanie krawędzi obiektów i poprawa ostrości.

1. Użyj filtra z maską Sobela do wykrywania krawędzi poziomych, pionowych i ukośnych.
2. Zaobserwuj działanie Laplasjanu do wyostrzania szczegółów.
3. Zbadaj działanie filtrów typu „unsharp masking” i „high boost”.

In [ ]:
```
import matplotlib.pyplot as plt
import tifffile as tiff
import cv2 as cv
from skimage.filters import sobel,laplace,unsharp_mask,gaussian
```

In [36]:
```
# Załadowanie pliku .tiff
img_a = tiff.imread("src/circuitmask.tif")
img_b = cv.imread("src/testpat1.png")
img_c = tiff.imread("src/blurry-moon.tif")
img_d = tiff.imread("src/text-dipxe-blurred.tif")
```

Zadanie 1

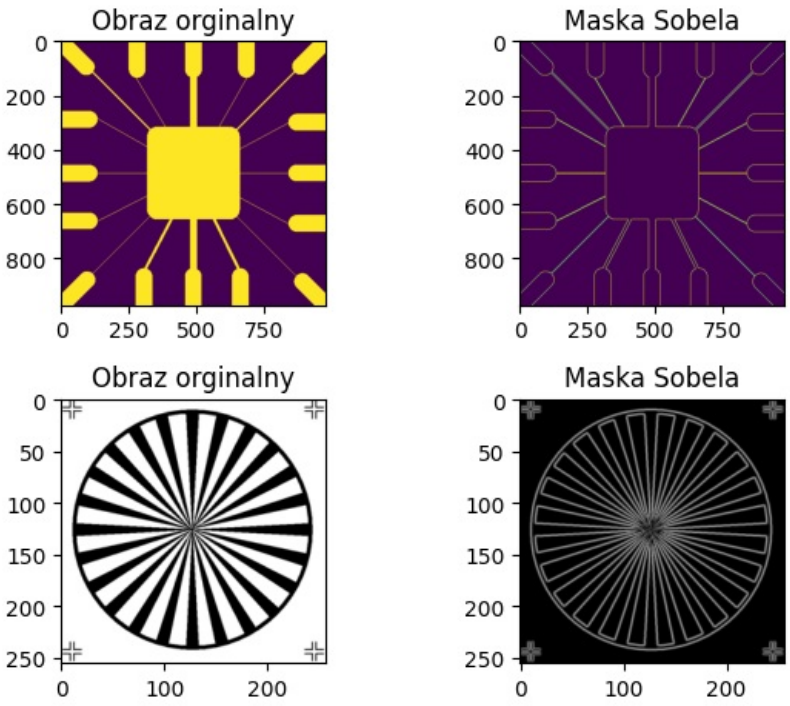
Użycie maski Sobela

In [37]:
```
plt.subplot(2,2,1)
plt.imshow(img_a)
plt.title("Obraz orginalny")

plt.subplot(2,2,2)
sob_img_a = sobel(img_a)
plt.imshow(sob_img_a)
plt.title("Maska Sobela")

plt.subplot(2,2,3)
plt.imshow(img_b)
plt.title("Obraz orginalny")

plt.subplot(2,2,4)
sob_img_b = sobel(img_b)
plt.imshow(sob_img_b)
plt.title("Maska Sobela")
plt.tight_layout()
plt.show()
```



Zadanie 2

Wykorzystanie Laplasjanu

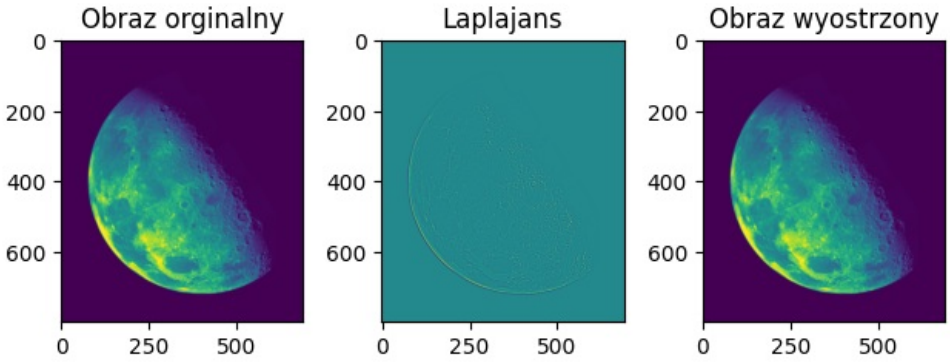
In [ ]:
```
plt.subplot(1,3,1)
plt.imshow(img_c)
```

```
plt.title("Obraz orginalny")

plt.subplot(1,3,2)
lap_img_c = laplace(img_c)
plt.imshow(lap_img_c)
plt.title("Laplasjan")

plt.subplot(1,3,3)
fin_img_c = img_c - lap_img_c
plt.imshow(fin_img_c)
plt.title("Obraz wyostrzony")

plt.tight_layout()
plt.show()
```



Zadanie 11

Wykorzystanie filtrów "unsharp masking" oraz "high boost"

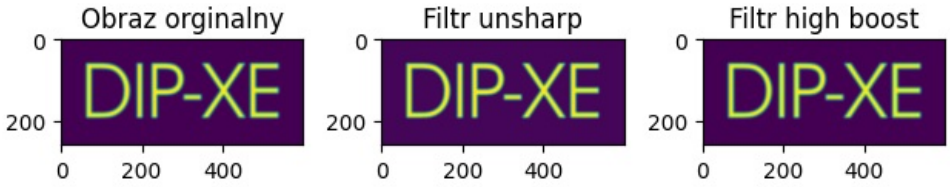
In [1]:
```
def highBoostFilter(img, k=1.5, sigma = 3):
    blur = gaussian(img,sigma=sigma)
    mask = k * img-blur
    return mask
```

In [48]:
```
plt.subplot(1,3,1)
plt.imshow(img_d)
plt.title("Obraz orginalny")

plt.subplot(1,3,2)
us_img_d = unsharp_mask(img_d)
plt.imshow(us_img_d)
plt.title("Filtr unsharp")

plt.subplot(1,3,3)
hb_img_d = highBoostFilter(img_d)
plt.imshow(hb_img_d)
plt.title("Filtr high boost")

plt.tight_layout()
plt.show()
```



Ćwiczenie 12

Naszym celem jest poprawa jakości obrazu za pomocą kolejnego stosowania różnych przekształceń i filtrów. Zastosuj złożone, wieloetapowe podej- ście do poprawy jakości przedstawione na wykładzie pt. „Filtracja w dziedzinie prze- strzennej”

In [11]:
```
import matplotlib.pyplot as plt
import tifffile as tiff
from numpy import emath
import skimage.morphology as morph
from skimage.filters import rank,sobel,laplace,unsharp_mask,gaussian
```

In [12]:
```
# Załadowanie pliku .tiff
img = tiff.imread("src/bonescan.tif")
```

1. Skan PET ciała człowieka
   - wysoki poziom szumów
   - dominacja ciemnych i jasnych poziomów szarości
2. Laplasjan obrazu 1. z maską 3 × 3
   - obraz przeskalowano do zakresu [0, 255]
3. Suma obrazów 1. i 2.
   - uwydatnienie drobnych szczegółów
   - wciąż zauważamy spory poziom szumów
4. Gradient Sobela obrazu 1.

   M (x, y) ≈ |gx| + |gy|
   - uwydatnienie brzegów

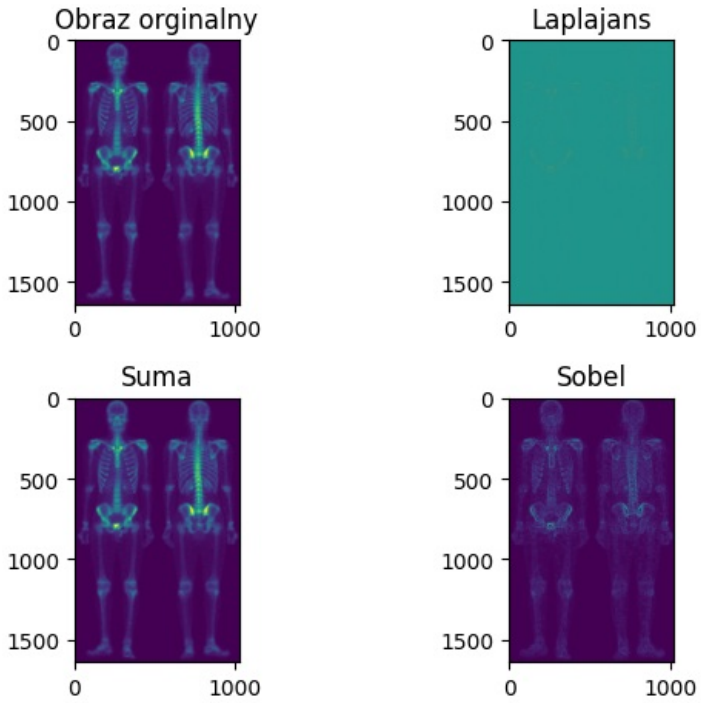
In [13]:
```
plt.subplot(2,2,1)
plt.imshow(img)
plt.title("Obraz orginalny")

plt.subplot(2,2,2)
lap_img = laplace(img)
plt.imshow(lap_img)
plt.title("Laplajans")

plt.subplot(2,2,3)
sum_img = img + lap_img
plt.imshow(sum_img)
plt.title("Suma")

plt.subplot(2,2,4)
sob_img = sobel(img)
plt.imshow(sob_img)
plt.title("Sobel")

plt.tight_layout()
plt.show()
```



5. Filtracja uśredniająca z maską 5 × 5 obrazu 4.
    - redukcja szumu uwydatnionego przez laplasjan
6. loczyn obrazu 5. i laplasjanu 2.
7. Suma 1. i 6.
8. Transformacja potęgowa 7., c = 1, γ = 0,5

    s = cr^γ
    - zwiększenie kontrastu

In [15]:
```
plt.subplot(2,2,1)
mean_img = rank.mean(img,morph.footprint_rectangle((5,5)))
plt.imshow(img)
plt.title("Filtracja uśredniająca")

plt.subplot(2,2,2)
ilo_img = mean_img * lap_img
plt.imshow(ilo_img)
plt.title("Iloczyn")

plt.subplot(2,2,3)
sum2_img = img + ilo_img
plt.imshow(sum2_img)
plt.title("Suma")

plt.subplot(2,2,4)
fin_img = ( sum2_img** 0.5 )
plt.imshow(fin_img)
plt.title("Efekt końcowy")

plt.tight_layout()
plt.show()
```

```
/tmp/ipykernel_21216/3351969261.py:17: RuntimeWarning: invalid value encountered in sqrt
  fin_img = ( sum2_img** 0.5 )
```