Zadanie 5

CYFROWE PRZETWARZANIE SYGNAŁÓW I OBRAZÓW
Przetwarzanie i analiza sygnału EKG

Kasper Radom 264023
Maciej Szymczak 263978

1

Importowanie niezbędnych bibliotek i deklaracja funkcji wczytującej obraz

In [6]:
```
%matplotlib ipympl
import skimage as ski
import matplotlib.pyplot as plt
from ipywidgets import interact, widgets
from matplotlib.gridspec import GridSpec
from matplotlib.widgets import SpanSelector

image = ski.io.imread('chest-xray.tif')
def readImage(imageName):
    global image
    image = ski.io.imread(imageName)


def showImage(image):
    plt.figure()
    ski.io.imshow(image)
    plt.title(dropdown.value[:-4])
    ski.io.show()
```

Wybór obrazu do wyświetlenia

In [7]:
```
options = ['aerial_view.tif', 'blurry-moon.tif', 'bonescan.tif', 'cboard_pepper_only.tif', 'cboard_salt_only.tif', 'cboard_salt_pepper.tif', 'characters_test_pattern.tif', 'chest-xray.tif',


dropdown = widgets.Dropdown(
    options=options,
    description='Wybierz plik:',
)

output = widgets.interactive_output(readImage, {'imageName': dropdown})
display(dropdown, output)

@interact(imageName = dropdown)
def closeAll(imageName):
    plt.close("all")
```

```
Dropdown(description='Wybierz plik:', options=('aerial_view.tif', 'blurry-moon.tif', 'bonescan.tif', 'cboard_p…
Output()
interactive(children=(Dropdown(description='Wybierz plik:', options=('aerial_view.tif', 'blurry-moon.tif', 'bo…
```

2

Wyświetlanie wykresu zmian poziomu szarości.
Wykres wyświetlany jest wzdłóż osi posiomej.
Znajduje się on na obrazie w celu lepszego wrażenia wizualnego przy porównywaniu wykresu z obrazem.
Czerwona linia pokazuje brany pod uwagę fragment obrazu

In [12]:
```
x = 300

#@interact(imageName = dropdown)
def showGreyScale(imageName):
    readImage(imageName)
    line_values = image[x, :]


    plt.figure()
    plt.plot(line_values)
    plt.plot(list(range(len(line_values))), [x] * len(line_values) , lw=0.2, color="red")
    ski.io.imshow(image)
    plt.xlabel('Pozycja na osi pionowej')
    plt.ylabel('Poziom szarości')
    plt.title('Zmiany poziomu szarości wzdłuż wybranej linii poziomej')
    plt.show()

showGreyScale("aerial_view.tif")
```

Figure

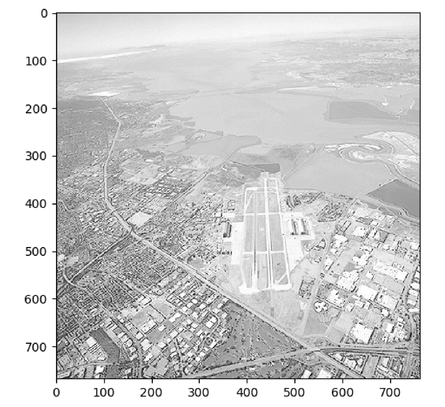
3

Wybór prostokątnego obrazu i jego zapis

In [9]:
```
plt.figure()
ski.io.imshow(image)

# Funkcja wywoływana po zmianie tekstu w polu tekstowym
def update_text(value):
    global newFileName
    newFileName = value

# Tworzenie interaktywnego pola tekstowego
text_widget = widgets.Text(value='NowyPlik.png', description='Nazwa pliku: ')
text_output = widgets.interactive_output(update_text, {'value': text_widget})

# Wyświetlenie pola tekstowego i wykresu
display(text_widget, text_output)
```

```
Text(value='NowyPlik.png', description='Nazwa pliku: ')
Output()
```

Figure



In [10]:
```
plt.savefig(newFileName)
```

In [ ]:

Zadanie 6

CYFROWE PRZETWARZANIE SYGNAŁÓW I OBRAZÓW
Przetwarzanie i analiza sygnału EKG

Kasper Radom 264023
Maciej Szymczak 263978

In [1]:
```
%matplotlib ipympl
import numpy as np
import skimage as ski
import matplotlib.pyplot as plt
from ipywidgets import interact, widgets
from skimage import filters
from skimage.morphology import disk
from skimage.filters import rank, laplace, sobel, gaussian
from skimage.exposure import rescale_intensity
from IPython.display import display
from matplotlib.widgets import RectangleSelector

image = ski.io.imread('chest-xray.tif')
```

deklaracja funkcji wczytującej obraz

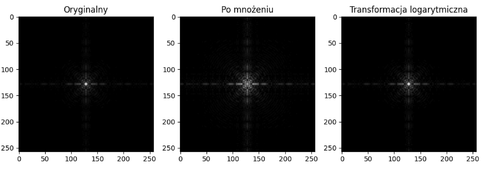
In [2]:
```
def readImage(imageName):
    global image
    image = ski.io.imread(imageName)


def showImage():
    plt.figure()
    ski.io.imshow(image)
    plt.title(dropdown.value[:-4])
    ski.io.show()
```

In [3]:
```
def multiply_image(image, constant):
    """Mnoży każdy piksel obrazu przez stałą."""
    # Używamy funkcji np.clip, aby upewnić się, że wartości pozostają w dopuszczalnym zakresie.
    return np.clip(image * constant, 0, 255).astype(np.uint8)


def logarithmic_transformation(image, constant):
    """Stosuje transformację logarytmiczną do obrazu."""
    # Skalujemy obraz do zakresu [0, 1], stosujemy transformację, a następnie skalujemy z powrotem.
    normalized_image = image / 255
    transformed_image = constant * np.log(1 + normalized_image)
    return np.clip(transformed_image * 255, 0, 255).astype(np.uint8)


image = ski.io.imread('spectrum.tif')  # Załaduj obraz
multiplied_image = multiply_image(image, 2)  # Przykład mnożenia przez stałą
log_transformed_image = logarithmic_transformation(image, 1)  # Przykład transformacji logarytmicznej
```

Wyświetlanie obrazów

In [4]:
```
# ORYGINALNY
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.imshow(image, cmap='gray')
plt.title('Oryginalny')

# a) PO PRZEMNOŻENIU
plt.subplot(1, 3, 2)
plt.imshow(multiplied_image, cmap='gray')
plt.title('Po mnożeniu')

# b) PO TRANSFORMACJI LOGARYTMICZNEJ
plt.subplot(1, 3, 3)
plt.imshow(log_transformed_image, cmap='gray')
plt.title('Transformacja logarytmiczna')
plt.show()
```

Figure



In [5]:
```
def contrast_adjustment(image, m, e):
    """Zmienia dynamikę skali szarości (kontrast) obrazu."""
    # Przekształcenie kontrastu
    image_float = image.astype(np.float32) / 255.0
    transformed_image = 1 / (1 + (m / (image_float + np.finfo(float).eps)) ** e)
    return np.clip(transformed_image * 255, 0, 255).astype(np.uint8)


# Wyświetlanie wykresu funkcji transformacji kontrastu
def plot_transformation_curve(m, e):
    r = np.linspace(0, 1, 256)
    T_r = 1 / (1 + (m / (r + np.finfo(float).eps)) ** e)
    plt.figure()
    plt.plot(r, T_r, label=f'm={m}, e={e}')
    plt.title('Wykres funkcji transformacji kontrastu')
    plt.xlabel('Wartość wejściowa r')
    plt.ylabel('Wartość wyjściowa T(r)')
    plt.legend()
    plt.show()
```

Eksperymenty z różnymi wartościami parametrów m i e

In [6]:
```
m, e = 0.45, 8
```

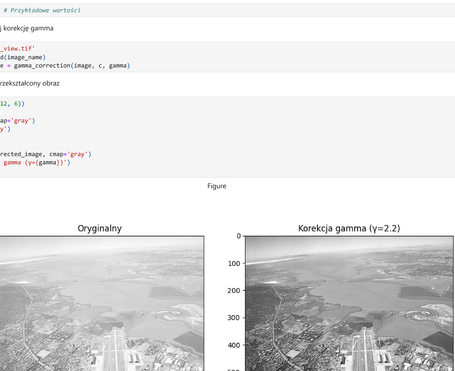
Załaduj obraz i wykonaj przekształcenie

In [7]:
```
image_name = 'einstein-low-contrast.tif'
image = ski.io.imread(image_name)
adjusted_image = contrast_adjustment(image, m, e)
```

Wyświetl oryginalny i przekształcony obraz

In [8]:
```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Oryginalny')

plt.subplot(1, 2, 2)
plt.imshow(adjusted_image, cmap='gray')
plt.title('Po zmianie kontrastu')
plt.show()
```

Figure



Wyświetl wykres funkcji transformacji

In [9]:
```
plot_transformation_curve(m, e)
```

Figure

In [10]:
```
def gamma_correction(image, c, gamma):
    """Stosuje korekcję gamma do obrazu."""
    # Normalizujemy obraz do zakresu [0, 1]
    normalized_image = image.astype(np.float32) / 255.0
    # Stosujemy korekcję gamma
    corrected_image = c * (normalized_image ** gamma)
    # Skalujemy z powrotem do zakresu [0, 255] i konwertujemy do typu uint8
    return np.clip(corrected_image * 255, 0, 255).astype(np.uint8)
```

Parametry dla przykładu korekcji gamma

In [11]:
```
c, gamma = 1.0, 2.2  # Przykładowe wartości
```

Załaduj obraz i wykonaj korekcję gamma

In [12]:
```
image_name = 'aerial_view.tif'
image = ski.io.imread(image_name)
gamma_corrected_image = gamma_correction(image, c, gamma)
```

Wyświetl oryginalny i przekształcony obraz

In [13]:
```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Oryginalny')

plt.subplot(1, 2, 2)
plt.imshow(gamma_corrected_image, cmap='gray')
plt.title(f'Korekcja gamma (γ={gamma})')
plt.show()
```

Figure

In [ ]:
```
```

Zadanie 7

CYFROWE PRZETWARZANIE SYGNAŁÓW I OBRAZÓW
Przetwarzanie i analiza sygnału EKG

Kasper Radom 264023
Maciej Szymczak 263978

Importowanie niezbędnych bibliotek i deklaracja funkcji wczytującej obraz

In [5]:
```
%matplotlib ipympl
import skimage as ski
import matplotlib.pyplot as plt
from ipywidgets import interact, widgets
from matplotlib.gridspec import GridSpec
from matplotlib.widgets import SpanSelector

image = ski.io.imread('chest-xray.tif')
def readImage(imageName):
    global image
    image = ski.io.imread(imageName)


def showImage(image):
    plt.figure()
    ski.io.imshow(image)
    plt.title(dropdown.value[:-4])
    ski.io.show()
```

Wybór obrazu

In [6]:
```
options = ['chest-xray.tif', 'aerial_view.tif', 'blurry-moon.tif', 'bonescan.tif', 'cboard_pepper_only.tif', 'cboard_salt_only.tif', 'cboard_salt_pepper.tif', 'characters_test_pattern.tif',


dropdown = widgets.Dropdown(
    options=options,
    description='Wybierz plik:',
)

output = widgets.interactive_output(readImage, {'imageName': dropdown})
display(dropdown, output)

@interact(imageName = dropdown)
def closeAll(imageName):
    plt.close("all")
```

```
Dropdown(description='Wybierz plik:', options=('chest-xray.tif', 'aerial_view.tif', 'blurry-moon.tif', 'bonesc…
Output()
interactive(children=(Dropdown(description='Wybierz plik:', options=('chest-xray.tif', 'aerial_view.tif', 'blu…
```

Wyświetlanie wybranego obrazu

In [9]:
```
#@interact(imageName = dropdown)
#def showOriginImage(imageName):
imageName = "chest-xray.tif"

readImage(imageName)
showImage(image)
```

Figure

Tworzenie i wyświetlanie histogramu obrazu

In [10]:
```
#@interact(imageName = dropdown)
#def equalize(imageName):
readImage(imageName)
imageEqualized = ski.exposure.equalize_hist(image)
showImage(imageEqualized)
```

Figure

In [ ]:
```

```

Zadanie 8

CYFROWE PRZETWARZANIE SYGNAŁÓW I OBRAZÓW
Przetwarzanie i analiza sygnału EKG

Kasper Radom 264023
Maciej Szymczak 263978

In [1]:
```
%matplotlib ipympl
import numpy as np
import skimage as ski
import matplotlib.pyplot as plt
from ipywidgets import interact, widgets
from skimage import filters
from skimage.morphology import disk
from skimage.filters import rank, laplace, sobel, gaussian
from skimage.exposure import rescale_intensity
from IPython.display import display
from matplotlib.widgets import RectangleSelector

image_name = 'hidden-symbols.tif'
image = ski.io.imread(image_name)
```

Lokalne wyrównywanie histogramu

In [2]:
```
def local_histogram_equalization(image, mask_size):
    # Stwórz strukturujący element (maskę)
    selem = disk(mask_size)
    # Zastosuj lokalne wyrównywanie histogramu
    equalized_image = rank.equalize(image, selem)
    return equalized_image
```

Poprawa jakości oparta na lokalnych statystykach

In [3]:
```
def local_statistical_enhancement(image, mask_size, C, k0, k1, k2, k3):
    selem = disk(mask_size)

    # Oblicz lokalne średnie i odchylenie standardowe
    local_mean = rank.mean(image, selem).astype('float') / 255.0
    local_mean_sq = rank.mean(image ** 2, selem).astype('float') / 255.0
    local_var = np.clip(local_mean_sq - local_mean ** 2, 0, None)
    # local_stddev = np.std(image, selem=selem).astype('float')
    local_stddev = np.sqrt(local_var)

    global_mean = np.mean(image)
    global_stddev = np.std(image)

    # Przekształć obraz według lokalnych statystyk

    enhanced_image = np.zeros_like(image, dtype='float')
    mask1 = (local_mean <= k0 * global_mean) & (local_stddev <= k1 * global_stddev)
    mask2 = (local_stddev > k2 * global_stddev)
    mask3 = (local_stddev <= k3 * global_stddev)

    enhanced_image[mask1] = C * np.log1p(image[mask1])
    enhanced_image[mask2 & ~mask1] = C * np.sqrt(image[mask2 & ~mask1])
    enhanced_image[mask3 & ~mask1 & ~mask2] = image[mask3 & ~mask1 & ~mask2]

    # Normalizacja obrazu do zakresu 0-255
    enhanced_image = ski.exposure.rescale_intensity(enhanced_image, in_range=(0, np.max(enhanced_image)))

    return enhanced_image
```

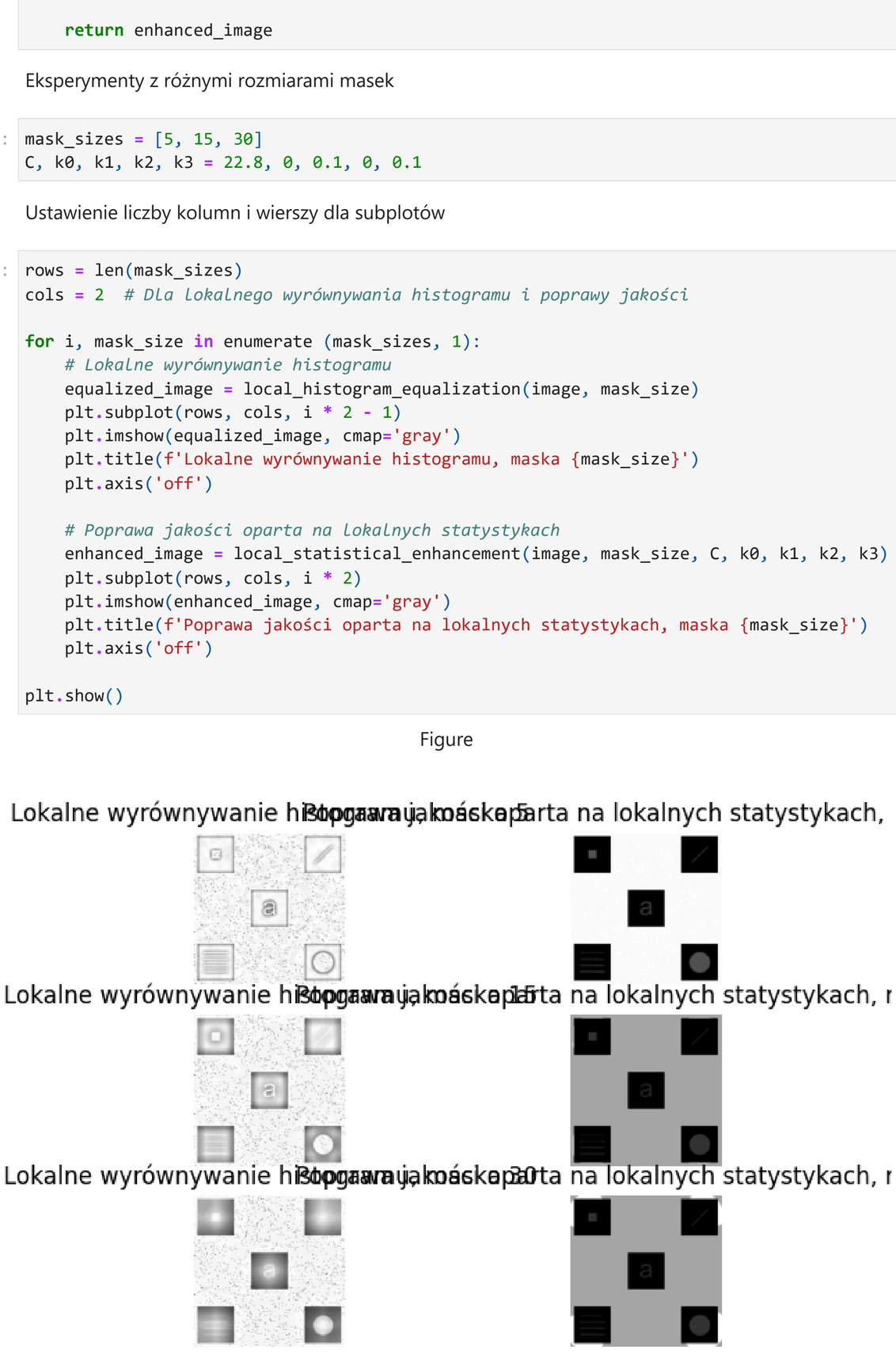
Eksperymenty z różnymi rozmiarami masek

In [4]:
```
mask_sizes = [5, 15, 30]
C, k0, k1, k2, k3 = 22.8, 0, 0.1, 0, 0.1
```

Ustawienie liczby kolumn i wierszy dla subplotów

In [5]:
```
rows = len(mask_sizes)
cols = 2  # Dla lokalnego wyrównywania histogramu i poprawy jakości

for i, mask_size in enumerate (mask_sizes, 1):
    # Lokalne wyrównywanie histogramu
    equalized_image = local_histogram_equalization(image, mask_size)
    plt.subplot(rows, cols, i * 2 - 1)
    plt.imshow(equalized_image, cmap='gray')
    plt.title(f'Lokalne wyrównywanie histogramu, maska {mask_size}')
    plt.axis('off')

    # Poprawa jakości oparta na lokalnych statystykach
    enhanced_image = local_statistical_enhancement(image, mask_size, C, k0, k1, k2, k3)
    plt.subplot(rows, cols, i * 2)
    plt.imshow(enhanced_image, cmap='gray')
    plt.title(f'Poprawa jakości oparta na lokalnych statystykach, maska {mask_size}')
    plt.axis('off')

plt.show()
```

Figure



In [ ]:
```
```

Zadanie 9

CYFROWE PRZETWARZANIE SYGNAŁÓW I OBRAZÓW
Przetwarzanie i analiza sygnału EKG

Kasper Radom 264023
Maciej Szymczak 263978

Importowanie niezbędnych bibliotek i deklaracja funkcji wczytującej obraz

In [1]:
```
%matplotlib ipympl
import numpy as np
import skimage as ski
import matplotlib.pyplot as plt
from ipywidgets import interact, widgets
from matplotlib.gridspec import GridSpec
from matplotlib.widgets import SpanSelector

image = ski.io.imread('cboard_salt_pepper.tif')
def readImage(imageName):
    global image
    image = ski.io.imread(imageName)


def showImage(image):
    plt.figure()
    ski.io.imshow(image)
    plt.title(dropdown.value[:-4])
    ski.io.show()
```

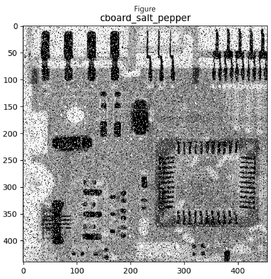
Wybór obrazu

In [2]:
```
options = ['cboard_salt_pepper.tif', 'chest-xray.tif', 'aerial_view.tif', 'blurry-moon.tif', 'bonescan.tif', 'cboard_pepper_only.tif', 'cboard_salt_only.tif', 'characters_test_pattern.tif',


dropdown = widgets.Dropdown(
    options=options,
    description='Wybierz plik:',
)

output = widgets.interactive_output(readImage, {'imageName': dropdown})
display(dropdown, output)

@interact(imageName = dropdown)
def closeAll(imageName):
    plt.close("all")
```

```
Dropdown(description='Wybierz plik:', options=('cboard_salt_pepper.tif', 'chest-xray.tif', 'aerial_view.tif', …
Output()
interactive(children=(Dropdown(description='Wybierz plik:', options=('cboard_salt_pepper.tif', 'chest-xray.tif…
```

Wyświetlanie wybranego obrazu

In [9]:
```
#@interact(imageName = dropdown)
#def showOriginImage(imageName):
imageName = "cboard_salt_pepper.tif"
readImage(imageName)
showImage(image)
```



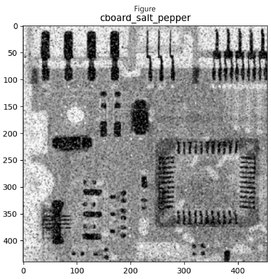
a)

Zastosowanie filtra o masce:

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

Filtr uśredniający

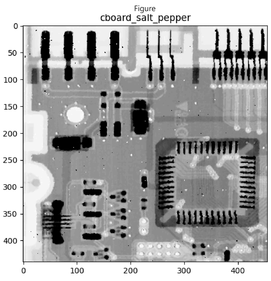
In [10]:
```
#@interact(imageName = dropdown)
#def filterMean(imageName):
readImage(imageName)
filtered_image = ski.filters.rank.mean(image, footprint=np.array([[1,1,1],[1,1,1],[1,1,1]]))
showImage(filtered_image)
```



b)

Nieliniowy filtr medianowy

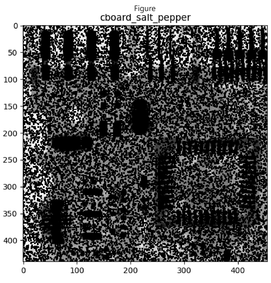
In [11]:
```
#@interact(imageName = dropdown)
#def filterMedian(imageName):
readImage(imageName)
filtered_image2 = ski.filters.rank.median(image)
showImage(filtered_image2)
```



c)

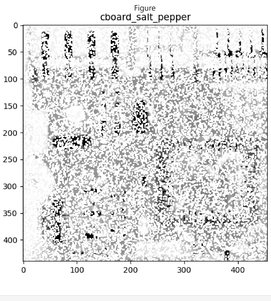
Filtr minimum

In [12]:
```
#@interact(imageName = dropdown)
#def filterMinimum(imageName):
readImage(imageName)
filtered_image2 = ski.filters.rank.minimum(image, footprint=np.array([[1,1,1],[1,1,1],[1,1,1]]))
showImage(filtered_image2)
```



Filtr maksimum

In [13]:
```
#@interact(imageName = dropdown)
#def filterMaximum(imageName):
readImage(imageName)
filtered_image2 = ski.filters.rank.maximum(image, footprint=np.array([[1,1,1],[1,1,1],[1,1,1]]))
showImage(filtered_image2)
```



In [ ]:
```
```

Zadanie 10

CYFROWE PRZETWARZANIE SYGNAŁÓW I OBRAZÓW
Przetwarzanie i analiza sygnału EKG

Kasper Radom 264023
Maciej Szymczak 263978

In [1]:
```
%matplotlib ipympl
import numpy as np
import skimage as ski
import matplotlib.pyplot as plt
from ipywidgets import interact, widgets
from skimage import filters
from skimage.morphology import disk
from skimage.filters import rank, laplace, sobel, gaussian
from skimage.exposure import rescale_intensity
from IPython.display import display
from matplotlib.widgets import RectangleSelector

image_name = 'characters_test_pattern.tif'
image = ski.io.imread(image_name)
```

Lista rozmiarów masek do eksperymentów

In [2]:
```
mask_sizes = [3, 5, 9, 15]
```

Filtracja uśredniająca

In [3]:
```
def apply_averaging_filter(image, mask_size):
    return filters.rank.mean(image, np.ones((mask_size, mask_size)))
```

Filtracja gaussowska

In [4]:
```
def apply_gaussian_filter(image, sigma):
    return filters.gaussian(image, sigma)
```

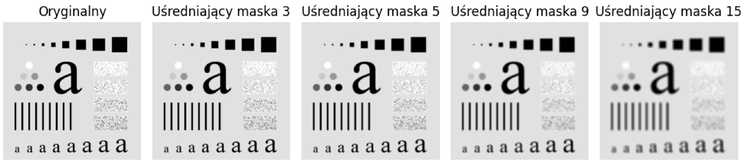
Wyświetlanie wyników

In [8]:
```
def display_filtered_images(image, filter_func, mask_sizes, filter_name):
    fig, axes = plt.subplots(1, len(mask_sizes) + 1, figsize=(10, 10), sharex=True, sharey=True)
    ax = axes.ravel()
    ax[0].imshow(image, cmap='gray')
    ax[0].set_title('Oryginalny')

    for i, mask_size in enumerate(mask_sizes, 1):
        filtered_image = filter_func(image, mask_size)
        ax[i].imshow(filtered_image, cmap='gray')
        ax[i].set_title(f'{filter_name} maska {mask_size}')

    for a in ax:
        a.axis('off')

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

Wyświetl obrazy po filtracji uśredniającej

In [9]:
```
display_filtered_images(image, apply_averaging_filter, mask_sizes, 'Uśredniający')
```

Figure



Wyświetl obrazy po filtracji gaussowskiej sigma zamiast rozmiarów masek, ponieważ filtr gaussowski jest parametryzowany przez sigma

In [10]:
```
sigmas = [1, 2, 3, 5]
display_filtered_images(image, apply_gaussian_filter, sigmas, 'Gaussowski')
```

Figure

In [ ]:
```
```

Zadanie 11

CYFROWE PRZETWARZANIE SYGNAŁÓW I OBRAZÓW
Przetwarzanie i analiza sygnału EKG

Kasper Radom 264023
Maciej Szymczak 263978

Importowanie niezbędnych bibliotek i deklaracja funkcji wczytującej obraz

In [1]:
```
%matplotlib ipympl
import numpy as np
import skimage as ski
import matplotlib.pyplot as plt
from ipywidgets import interact, widgets
from matplotlib.gridspec import GridSpec
from matplotlib.widgets import SpanSelector

image = ski.io.imread('circuitmask.tif')
def readImage(imageName):
    global image
    image = ski.io.imread(imageName)


def showImage(image):
    plt.figure()
    #ski.io.imshow(image)
    plt.imshow(image, cmap='gray')
    plt.title(dropdown.value[:-4])
    ski.io.show()

def showBoth(image1, image2):
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.imshow(image1, cmap='gray')
    plt.title('Oryginalny obraz')
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.imshow(image2, cmap='gray')
    plt.axis('off')

    plt.show()
```

Wybór obrazu

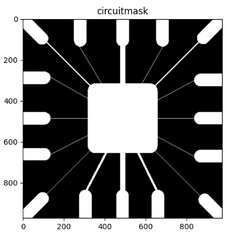
In [2]:
```
options = ['circuitmask.tif', 'cboard_salt_pepper.tif', 'chest-xray.tif', 'aerial_view.tif', 'blurry-moon.tif', 'bonescan.tif', 'cboard_pepper_only.tif', 'cboard_salt_only.tif', 'characters_


dropdown = widgets.Dropdown(
    options=options,
    description='Wybierz plik:',
)

output = widgets.interactive_output(readImage, {'imageName': dropdown})
display(dropdown, output)


@interact(imageName = dropdown)
def closeAll(imageName):
    plt.close("all")
```

```
Dropdown(description='Wybierz plik:', options=('circuitmask.tif', 'cboard_salt_pepper.tif', 'chest-xray.tif', …
Output()
interactive(children=(Dropdown(description='Wybierz plik:', options=('circuitmask.tif', 'cboard_salt_pepper.ti…
```

Wyświetlanie wybranego obrazu

In [10]:
```
#@interact(imageName = dropdown)
#def showOriginImage(imageName):
imageName = "circuitmask.tif"
readImage(imageName)
showImage(image)
```

Figure

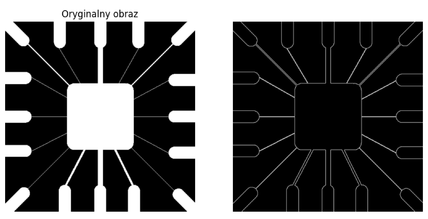


a)

Wykrywanie krawędzi (Sobel)

In [11]:
```
#@interact(imageName = dropdown)
#def filterSobel(imageName):
readImage(imageName)
filtredImage = ski.filters.sobel(image)
showBoth(image, filtredImage)
```

Figure



b)

Wyostrzanie laplasjanem

In [16]:
```
#@interact(imageName = dropdown)
#def filterLaplace(imageName):
imageName = "blurry-moon.tif"
readImage(imageName)
filtredImage = image + (-1) * ski.filters.laplace(image) ** 2
showBoth(image, filtredImage)
```

Figure

c)

Unsharpmask

In [17]:
```
#@interact(imageName = dropdown)
#def filterUnsharp(imageName):
imageName = "text-dipxe-blurred.tif"
readImage(imageName)
filtredImage = ski.filters.unsharp_mask(image, radius=5, amount=1)
showBoth(image, filtredImage)
```

Figure

In [18]:
```
#@interact(imageName = dropdown)
#def filterHighboost(imageName):
readImage(imageName)
filtredImage = ski.filters.unsharp_mask(image, radius=10, amount=4.5)
showBoth(image, filtredImage)
```

Figure

In [ ]:
```
```

Zadanie 12

CYFROWE PRZETWARZANIE SYGNAŁÓW I OBRAZÓW
Przetwarzanie i analiza sygnału EKG

Kasper Radom 264023
Maciej Szymczak 263978

In [16]:
```
%matplotlib ipympl
import numpy as np
import skimage as ski
import matplotlib.pyplot as plt
from ipywidgets import interact, widgets
from skimage import filters
from skimage.morphology import disk
from skimage.filters import rank, laplace, sobel, gaussian
from skimage.exposure import rescale_intensity
from IPython.display import display
from matplotlib.widgets import RectangleSelector

def readImage(imageName):
    global image
    image = ski.io.imread(imageName)
```

In [17]:
```
def filterLaplace(imageName):
    readImage(imageName)
    filteredImage = image + (-1) * ski.filters.laplace(image) ** 2
    return filteredImage

image_name = 'bonescan.tif'
image = ski.io.imread(image_name)
```

a) Oryginalny obraz b) Laplacjan obrazu (a)

In [18]:
```
laplacian_image = filterLaplace(image_name)
```

c) Suma obrazów (a) i (b)

In [19]:
```
sum_laplacian = image + laplacian_image
```

d) Gradiend Sobela obrazu (a)

In [20]:
```
sobel_image = sobel(image)
```

e) Filtracja uśredniająca z maską 5x5 obrazu (d)

In [21]:
```
avg_filter_image = gaussian(sobel_image, sigma=1)
```

f) Iloczyn obrazu (e) i laplasjanu (b)

In [22]:
```
mult_image = avg_filter_image * laplacian_image
```

g) Suma (a) i (f)

In [23]:
```
sum_image = image + mult_image
```

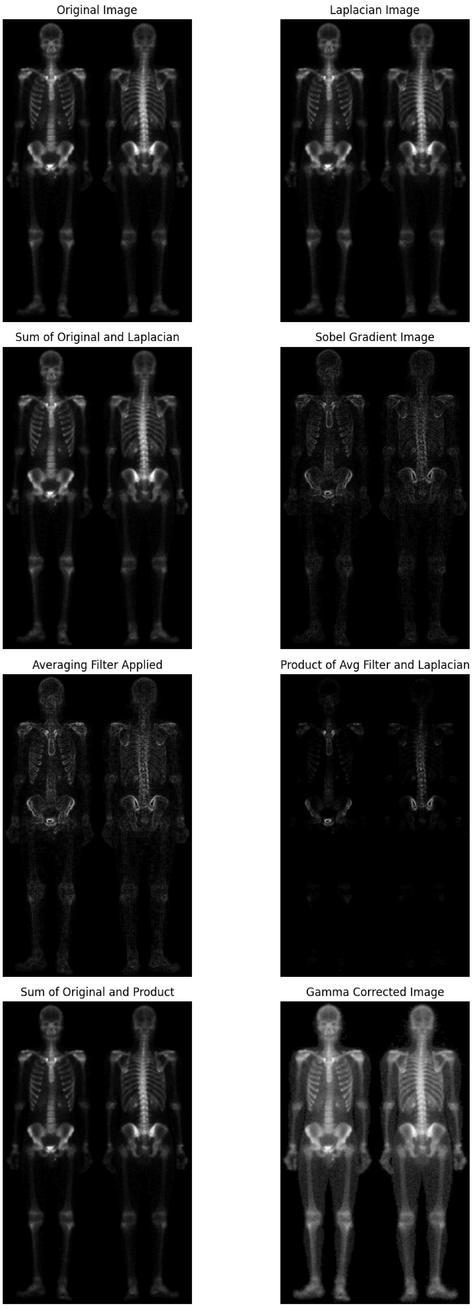
h) Transformacja potęgowa (g) Przycinanie wartości ujemnych do zera przed pierwiastkowaniem

In [24]:
```
sum_image_clipped = np.clip(sum_image, 0, None)
gamma_corrected = rescale_intensity(np.sqrt(sum_image_clipped))
```

Wyświetlanie wyników

In [25]:
```
fig, ax = plt.subplots(4, 2, figsize=(10, 20), sharex=True, sharey=True)
ax = ax.ravel()

# Wyostrzony obraz po dodaniu laplasjanu
ax[0].imshow(image, cmap='gray')
ax[0].set_title('Original Image')

ax[1].imshow(laplacian_image, cmap='gray')
ax[1].set_title('Laplacian Image')

ax[2].imshow(sum_laplacian, cmap='gray')
ax[2].set_title('Sum of Original and Laplacian')

ax[3].imshow(sobel_image, cmap='gray')
ax[3].set_title('Sobel Gradient Image')

ax[4].imshow(avg_filter_image, cmap='gray')
ax[4].set_title('Averaging Filter Applied')

ax[5].imshow(mult_image, cmap='gray')
ax[5].set_title('Product of Avg Filter and Laplacian')

ax[6].imshow(sum_image, cmap='gray')
ax[6].set_title('Sum of Original and Product')

ax[7].imshow(gamma_corrected, cmap='gray')
ax[7].set_title('Gamma Corrected Image')

for a in ax:
    a.axis('off')
plt.tight_layout()
plt.show()
```

Figure



In [ ]: