

Analiza obrazów

Sprawozdanie nr 1 z ćwiczeń laboratoryjnych

Mikołaj Marchewa

Wydział Fizyki i informatyki Stosowanej

Akademia Górnictwo-Hutnicza im. Stanisława Staszica w Krakowie

17 listopada 2020

1. Laboratorium 2

Celem drugich zajęć laboratoryjnych było zapoznanie się z podstawowymi pojęciami i algorytmami wykorzystywanyimi w analizie obrazów. Do naszych ćwiczeń wykorzystaliśmy poniższy obraz:



Rys. 1: Oryginał obrazu wykorzystywanego podczas zajęć.

Na początku unormalizowaliśmy nasz obraz, tak by przechowywać go w macierzy zmiennych typu *double*, co ułatwi nam kolejne wykonywane operacje:

```
im = double(im)/255;
```

Nasz obraz jest więc od tej pory przechowywany w macierzy o wymiarach: *wysokość x szerokość x 3 kanały zawierające informacje o danym kolorze na obrazie – RGB*. Mając tą wiedzę i wykorzystując możliwości narzędzia MATLAB możemy w łatwy sposób manipulować jedynie interesującymi nas kolorami na obrazie:

```
rim = im;
rim(:, :, [1 2]) = 0;
```

Efektem powyższej operacji jest pozbycie się wszystkich informacji (wyzerowanie zawartych w nich wartości) o kanałach R (czerwony) oraz G (zielony), co przedstawia Rys.2:



Rys. 2: Oryginał obrazu pozbawiony kolorów zielonego i czerwonego.

1.1 Konwersja do skali szarości

W celu uzyskania większej wydajności wykonywanych obliczeń, a także dalszej analizy obrazu poznaliśmy sposoby, które pozwalają nam na przekonwertowanie obrazu do skali szarości. Obraz taki przechowywany jest w macierzy dwuwymiarowej, której wymiary to: wysokość \times szerokość, a każda poszczególna interesująca nas wartość w macierzy będzie mieściła się w zakresie $[0;1]$, gdzie

- 0 - oznacza brak jasności piksela,
- 1 - oznacza maksymalną jasność piksela w danym punkcie na obrazie.

Znając powyższe informacje przekonwertowaliśmy Rys.1 do skali szarości wykorzystując tylko 2 kanał (zielony):

```
rim = im(:,:,2);
```

co dało efekt widoczny na Rys.3:



Rys. 3: Oryginał przekonwertowany do skali szarości z wykorzystaniem kanału 2.

Niestety konwertując nasz obraz wykorzystując jedynie pojedynczy kanał tracimy wiele cennych informacji, dlatego lepszym sposobem będzie wykorzystanie średniej arytmetycznej z trzech kanałów jako wartość jasności piksela w punkcie:

```
gim = mean(im,3);
```

widoczne na Rys.4:



Rys. 4: Konwersja do skali szarości z wykorzystaniem średniej.

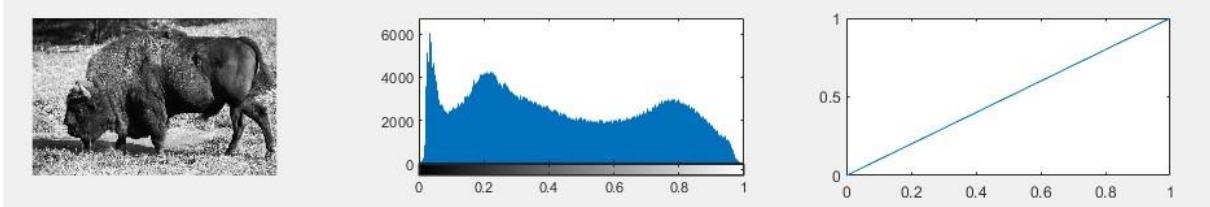
Innym sposobem przejścia na skalę szarości, który poznaliśmy w czasie zajęć było użycie modelu barw YUV (wykorzystywanego do konwersji obrazu w czasie przejścia z telewizji z czarno-białej na kolorową):

```
w = [.299, .587, .114]; % wektor wag  
w = permute(w, [1,3,2]); % zmiana wymiaru wektora wag  
gim = sum(im.*w, 3); % przejście na skalę szarości
```



Rys. 5: Konwersja do skali szarości z wykorzystaniem modelu barw YUV.

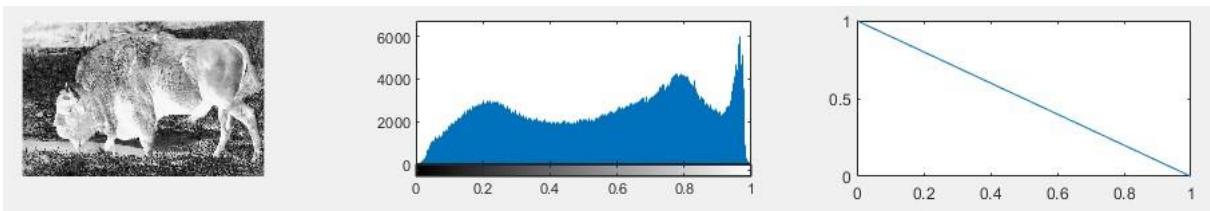
Powyższą konwersję z Rys.5 możemy realizować również za pomocą wbudowanej funkcji `rgb2gray`, co wykorzystamy do dalszej części laboratorium. Działanie metody `rgb2gray` wraz z odpowiadającymi jej wykresem histogramu oraz wykres przekształcenia posłużą nam jako punkt odniesienia do wykonywanych operacji na obrazie.



Rys. 6: Konwersja z użyciem metody `rgb2gray` wraz z histogramem i wykresem przekształcenia dla oryginalnego obrazu.

1.2 Inwersja

Pierwszą poznaną operacją dokonaną na naszym obrazie w skali szarości była inwersja. Polega ona na odwróceniu wartości każdego piksela tzn. tam gdzie piksel był jasny – stanie się on ciemny i analogicznie dla pikseli ciemnych. By wykonać tą operację wystarczy od wartości 1 odjąć wartość każdego piksela.



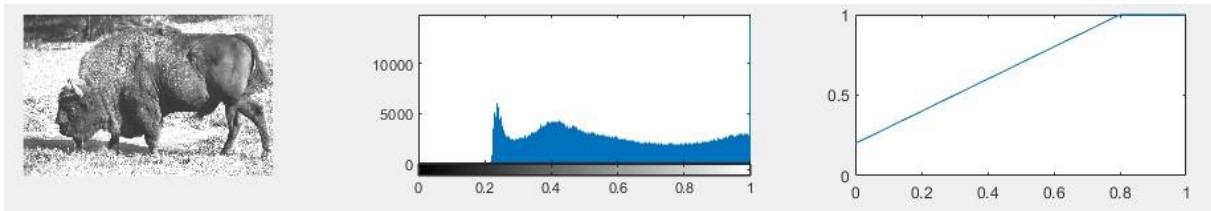
Rys. 7: Konwersja z użyciem metody `rgb2gray` oraz nałożoną inwersją wraz z histogramem i wykresem przekształcenia dla oryginalnego obrazu.

1.3 Zmiana jasności

By dokonać zmiany jasności obrazu wystarczy wykonać operację dodania wybranej przez nas wartości do każdego z pikseli. Wartość ta powinna mieścić się w przedziale $(-1;1)$, by nasz obraz nie zmienił się w jednokolorowy – czarny bądź biały. Dla przykładu wybieramy naszą wartość, o którą „przesuniemy” jasności każdego punktu na obrazie, na $+0.2$. Jednak jak już początkowo ustaliliśmy, interesujące nas wartości jasności pikseli muszą mieścić się w przedziale $[0;1]$, dlatego po przesunięciu normujemy nasz obraz. Wykonujemy więc kolejno operacje:

```
b = 0.2;
bim = gim + b;
bim(bim<0) = 0; % zabezpieczenie przed wyjściem poza przedział [0;1]
bim(bim>1) = 1; % zabezpieczenie przed wyjściem poza przedział [0;1]
```

a następnie przedstawiliśmy rezultaty:



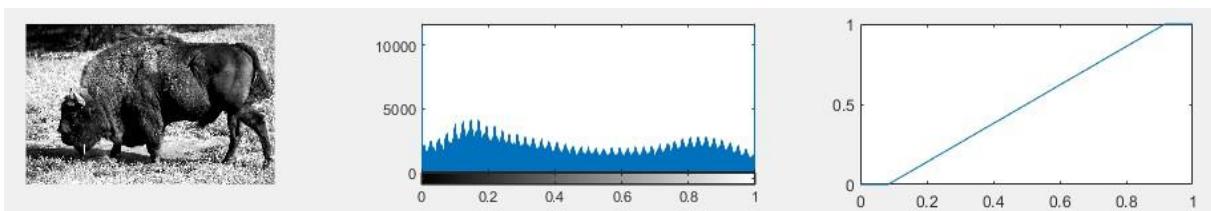
Rys. 8: Konwersja z użyciem metody `rgb2gray` i przesunięciem o wartość +0.2, wraz z histogramem i wykresem przekształcenia.

Teoria pokryła się z praktyką przez co uzyskaliśmy znaczaco rozjaśniony obraz, zaś histogram i wykres przesunięcia różnią się od punktu wyjściowego z Rys.6 o naszą zadaną wartość +0.2.

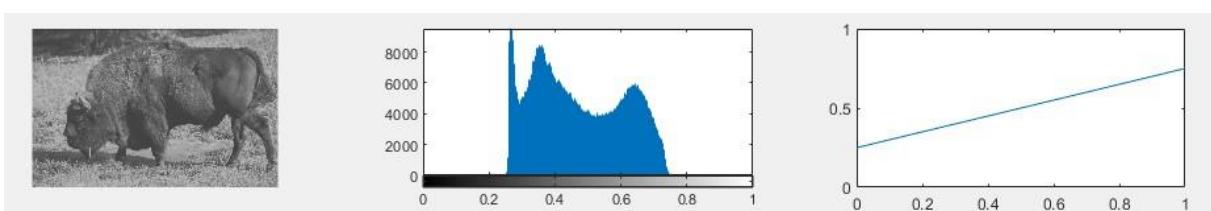
1.4 Zmiana kontrastu

Zmiana kontrastu obrazu w skali szarości odbywa się poprzez pomnożenie każdego piksela na obrazie o zadaną stałą wartość. Jednak by powyższa operacja dawała prawidłowe rezultaty konieczne jest początkowe przesunięcie obrazu o -0.5, a następnie po przemnożeniu przez stałą dodanie +0.5. Dzięki czemu mnożąc przez stałą nasz wykres uwypukli się, bądź wypłaszczy na swoich krańcach powiększając, bądź zmniejszając różnicę pomiędzy najjaśniejszymi i najciemniejszymi pikselami. Gdy wartość stałej mieści się w przedziale (0;1) nasz obraz straci na kontraste, zaś gdy wybierzemy stałą z przedziału (1;+∞) pozwoli nam to na uzyskanie przeciwnego efektu – zwiększenie się kontrastu.

```
c = 1.2; % stała
cim = (gim-0.5)*c+0.5; % zmiana kontrastu
cim(cim<0) = 0;
cim(cim>1) = 1;
```



Rys. 9: Konwersja z użyciem metody `rgb2gray` i zwiększenie kontrastu dla $c=1.2$, wraz z histogramem i wykresem przekształcenia.



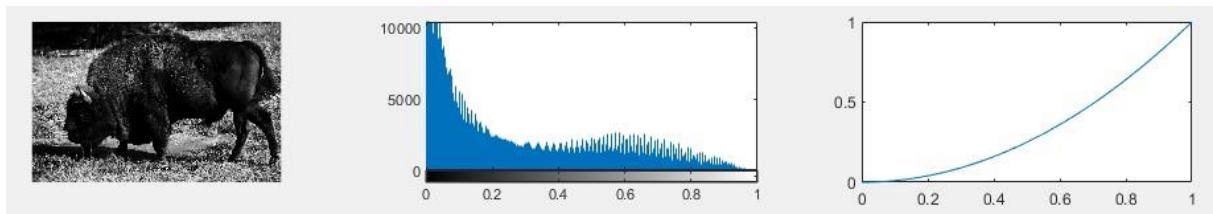
Rys. 10: Konwersja z użyciem metody `rgb2gray` i zmniejszenie kontrastu dla $c=0.5$, wraz z histogramem i wykresem przekształcenia.

Histogram z Rys.9 doskonale uwidacznia co dzieje podczas zwiększenia kontrastu – zwiększają się różnice pomiędzy wartościami pikseli przez co histogram staje się „poszarpany”, a wykres przesunięcia zwiększa kąt nachylenia. Przeciwne zjawiska możemy dostrzec na wykresach z Rys.10.

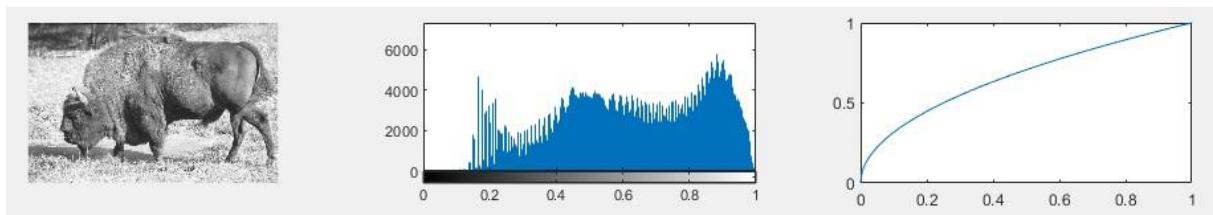
1.5 Korekcja gamma

Korekcja gamma obrazu polega na wykonaniu operacji potęgowania każdego piksela. Tym razem dodatkowo odwracamy wybraną stałą, by uzyskać intuicyjność wykonywanej operacji – gdy wybierzymy małe wartości stałej gamma z przedziału $(0;1)$, wówczas obraz powinien zmniejszyć swoją jasność. Gdy zaś wybierzymy wartość z przedziału $(1;+\infty)$, wtedy zyskam rozjaśnienie obrazu.

```
gamma = 0.5; % stała
gammaim = gim.^^(1/gamma); % korekcja gamma
gammaim(gammaim<0) = 0;
gammaim(gammaim>1) = 1;
```



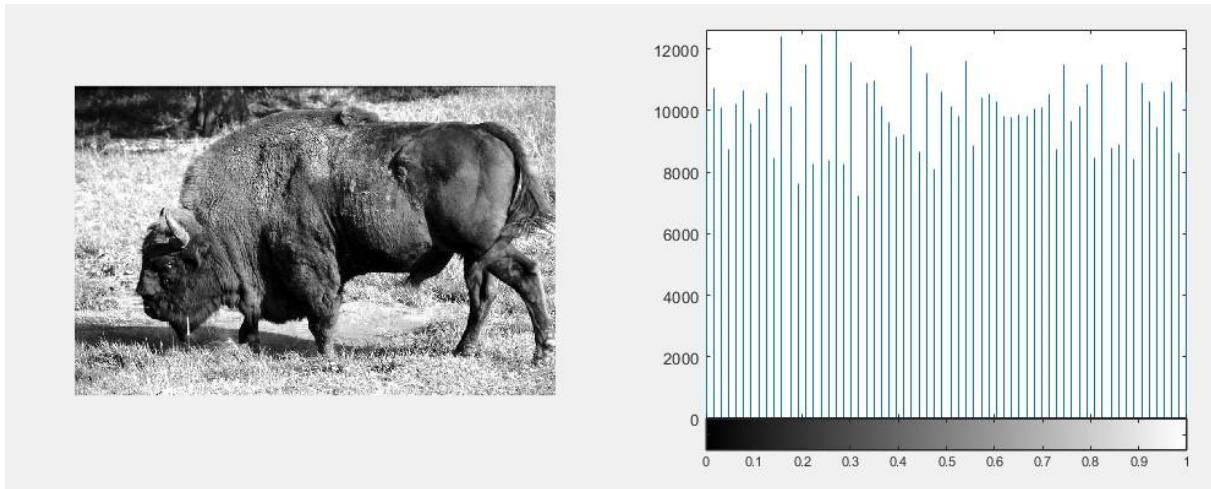
Rys. 11: Konwersja z użyciem metody `rgb2gray` i korekcja gamma dla $\text{gamma}=0.5$, wraz z histogramem i wykresem przekształcenia.



Rys. 12: Konwersja z użyciem metody `rgb2gray` i korekcja gamma dla $\text{gamma}=2$, wraz z histogramem i wykresem przekształcenia.

1.6 Normalizacja

Na zakończenie poznaliśmy funkcję `histeq`, pozwalającą na wzmacnianie kontrastu z pomocą wyrównania (normalizacji) histogramu.



Rys. 13: Konwersja z użyciem metody `rgb2gray` i normalizacja obrazu wraz z histogramem.

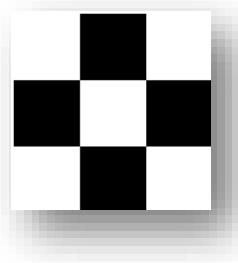
2. Laboratorium 3

W trakcie trwania tych ćwiczeń laboratoryjnych zapoznaliśmy się z podstawowymi filtrami, a także z operacjami niezbędnymi do przygotowania obrazu do nałożenia na niego filtru. Podobnie jak w laboratorium 2, jako punkt odniesienia używaliśmy przekonwertowanego do skali szarości zdjęcia żubra (Rys.6).

Na początku poznaliśmy pojęcie sąsiedztwa w kontekście analizy obrazów.

- Sąsiedztwo Moore'a – sąsiedztwo centralnego (środkowego) piksela z ośmioma otaczającymi go pikselami, które graniczą z nim krawędzią, bądź wierzchołkiem.
- Sąsiedztwo von Neumanna – sąsiedztwo centralnego piksela z czterema otaczającymi go pikselami, graniczącymi z tym pikselem jedynie krawędziami.

Dla intuicyjnego zrozumienia powyższych pojęć stworzyliśmy szachownice symbolizującą piksel centralny wraz z jego otoczeniem (Rys.14). Na potrzeby naszych zajęć wykorzystywaliśmy sąsiedztwo Moore'a do nakładania filtrów.



Rys. 14: Szachownica, której kwadraty symbolizują piksele na obrazie.

2.1 Filtr dolnoprzepustowy

Pierwszym poznanym filtrem był filtr dolnoprzepustowy, w którym każdy piksel zależy w równym stopniu od pikseli w jego sąsiedztwie i od piksela centralnego.

```
f = ones(7); % macierz 7x7 wypełniona jedynkami - filtr  
f = f/sum(f, 'all'); % normalizacja filtru  
fim = imfilter(gim, f); % nałożenie filtru na obraz
```

Macierz f jest więc macierzą identycznych wag (w tym przypadku waga będzie równa $\frac{1}{7^2}$, gdzie liczba 7 wynika z wymiaru macierzy), które sumują się do 1. Efekt nałożenia filtru na nasz obraz, z wykorzystaniem funkcji *imfilter* obrazuje Rys.15.



Rys. 15: Obraz oryginalny w skali szarości (po lewo) i ten sam obraz z nałożonym filtrem dolnoprzepustowym (po prawo).

Skutkiem wykonania tej operacji jest rozmycie i „zlagodzenie” granic pomiędzy ciemnymi i jasnymi częściami obrazu, a także pojawienie się ciemnej ramki, która wynika z faktu iż wartości pikseli wliczane do macierzy f wychodzące poza granice zdjęcia są równe o.

2.2 Filtr górnoprzepustowy

Kolejnym rodzajem filtru jaki mieliśmy okazję poznać na tym laboratorium był filtr górnoprzepustowy. Podobnie jak filtr dolnoprzepustowy wykorzystuje on do swojego działania macierz wag – sąsiedztwo pikseli. Jednak różnicą między tymi filtrami jest to jakie wagi wybieramy. Dla filtru górnoprzepustowego piksele otaczające piksel centralny będą miały wartość -1, zaś piksel centralny przyjmie wartość dodatnią taką, aby po zsumowaniu macierzy otrzymać 1.

```
f = -ones(3);  
f(2,2) = 9; % ustawienie wartości piksela centralnego  
fim = imfilter(gim, f);
```



Rys. 16: Obraz oryginalny w skali szarości (po lewo) i ten sam obraz z nałożonym filtrem górnoprzepustowym (po prawo).

Dzięki filtrowi górnoprzepustowemu uzyskujemy wyostrzenie obrazu – zwiększenie różnic pomiędzy pikselami centralnymi i sąsiednimi pikselami.

2.3 Połączenie filtru dolno- i górnoprzepustowego

Ciekawym efektem jest złączenie obu wyżej wymienionych filtrów. Kiedy najpierw zastosujemy filtr dolnoprzepustowy, a następnie górnoprzepustowy, wówczas ten pierwszy zmniejszy zaszumienie obrazu, a kolejny wyostrzy obraz. Dzięki takiej operacji uzyskamy przydatną w fotografii korekcję zdjęcia (Rys.17).



Rys. 17: Obraz oryginalny w skali szarości (po lewo) i ten sam obraz z nałożonym filtrem dolno- i górnoprzepustowym (po prawo).

2.4 Filtr medianowy

Filtr medianowy polega na utworzeniu „okna”, którym sprawdzamy wartości pikseli i ich otoczenia, po czym po posortowaniu tych wartości wybieramy wartość środkową – medianę. Zaletą tego filtra jest jego odporność na zakłócenia jednostkowe tzn. pojedyncze piksele znacząco wyróżniające się spośród swojego otoczenia. Na potrzeby zrealizowania filtra medianowego wykorzystaliśmy gotową funkcję *medfilt2*, zaś uzyskany rezultat przedstawiliśmy na Rys.18.

```
fim = medfilt2(gim);
```

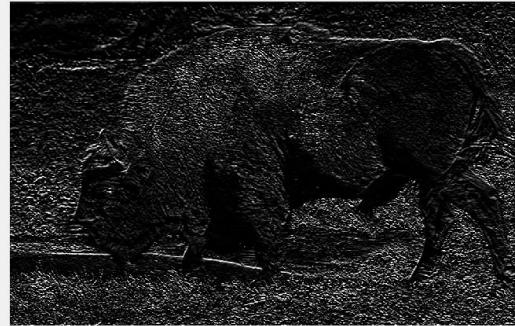


Rys. 18: Obraz oryginalny w skali szarości (po lewo) i ten sam obraz z nałożonym filtrem medianowym (po prawo).

2.5 Filtr Prewitt

Filtr Prewitt wykorzystywany jest by wykrywać na obrazie krawędzie poziome.

```
f = fspecial('prewitt');
fim = imfilter(gim, f);
```



Rys. 19: Obraz oryginalny w skali szarości (po lewo) i ten sam obraz z nałożonym filtrem Prewitt'a (po prawo).

2.6 Filtr Sobel

Podobnie jak filtr Prewitt, filtr Sobel pozwala na wykrycie krawędzi poziomych na obrazie.

```
f = fspecial('sobel');
fim = imfilter(gim, f);
```



Rys. 20: Obraz oryginalny w skali szarości (po lewo) i ten sam obraz z nałożonym filtrem Sobel'a (po prawo).

2.7 Binaryzacja

Binaryzacja obrazu oznacza przejście ze skali szarości do obrazu czarno-białego. Realizujemy ją poprzez zmianę wartości każdego z pikseli na 1 lub 0, w zależności od tego czy dany piksel jest większy, bądź mniejszy od wybranego progu – stałej.

Do wyznaczenia progu możemy wykorzystać histogram oryginalnego obrazu i określić go intuicyjnie, bądź skorzystać z gotowych funkcji takich jak *graythresh*.

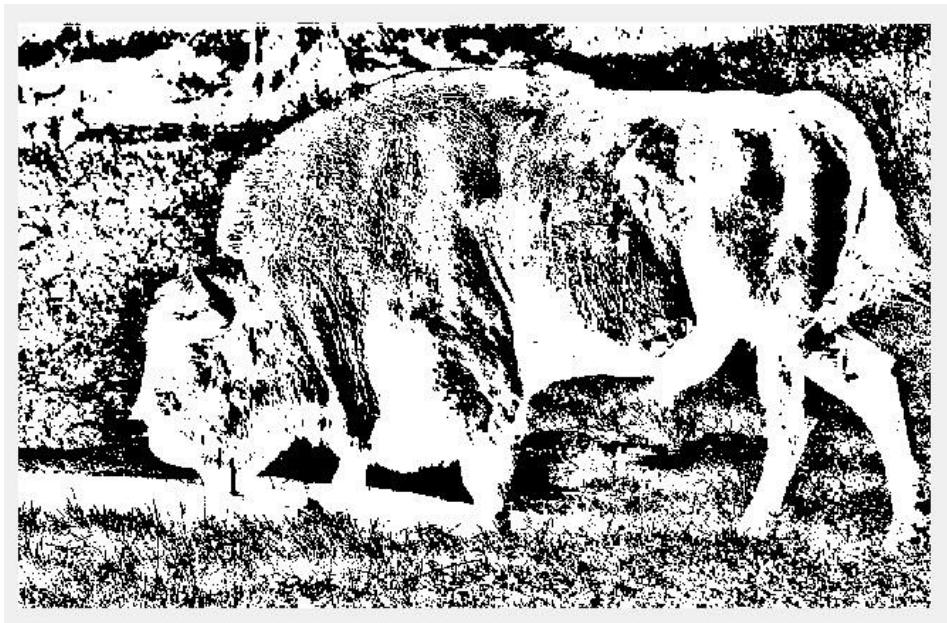
```
t = graythresh(gim);  
bim = imbinarize(gim, t);
```



Rys. 21: *Binaryzacja z wykorzystaniem progu z funkcji graythresh (po lewo) oraz binaryzacja z wybranym progiem na podstawie histogramu ustalonego na 0.6 (po prawo).*

W ogólności chcemy osiągnąć biały obiekt i czarne tło. Dlatego przy analizie większej ilości obrazów możemy używać funkcję *imbinarize* z wykorzystaniem metody ‘adaptive’, która pozwala na lokalne, adaptacyjne wyznaczenie progu (threshold).

```
bim = ~imbinarize(gim, 'adaptive');
```



Rys. 22: *Binaryzacja z wykorzystaniem progu z funkcji imbinarize wraz z metodą ‘adaptive’ i inwersją.*

2.8 Erozja

Erozja polega na sprawdzeniu otoczenia każdego piksela na obrazie w celu zdejmowania kolejnych warstw. Jeśli choć jeden piksel w sąsiedztwie, bądź piksel centralny przyjmuje wartość 0, wówczas punkt środkowy przyjmuje wartość 0. W przeciwnym przypadku piksel pozostaje biały.

```
bim =imerode(bim, ones(3)); % ones(3) – okno sprawdzające  
% sąsiedztwo piksela
```

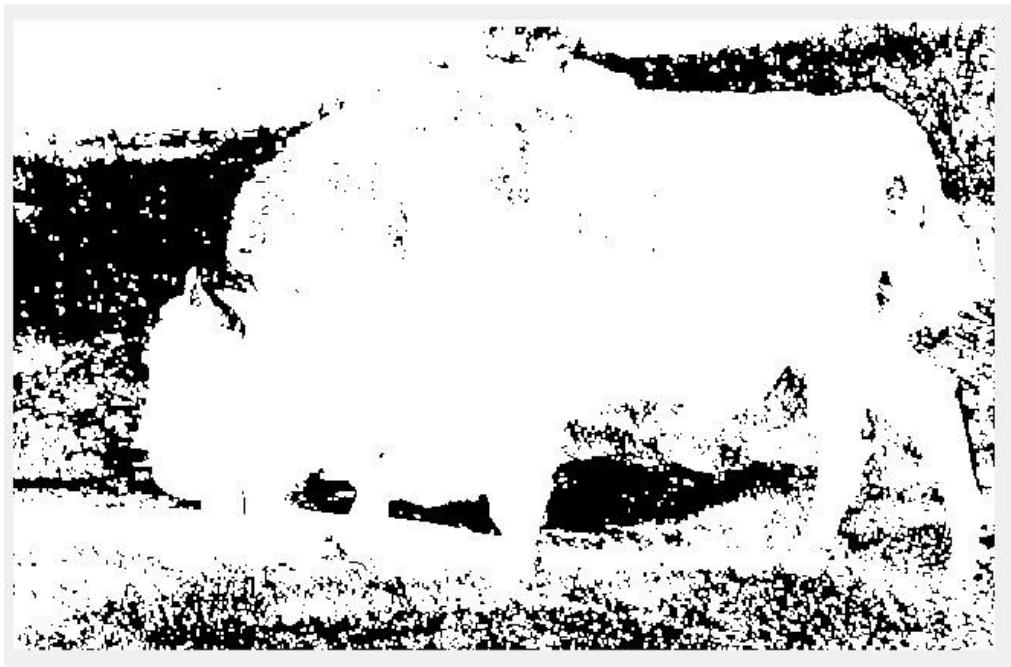


Rys. 23: Erozja obrazu.

2.9 Dylatacja

Dylatacja jest operacją analogiczną do erozji z tą różnicą, że dla dylatacji piksel centralny zmieniamy na wartość 1 jeśli jego otoczenie przyjmuje choć jedną wartość 1.

```
bim = imdilate(bim,ones(3)); % ones(3) – okno sprawdzające  
% sąsiedztwo piksela
```



Rys. 24: Dylatacja obrazu.

2.10 Operacja zamknięcia

Dokonanie następujących po sobie kolejno operacji dylatacji i erozji nazywamy operacją zamknięcia. Zastosowanie tej operacji morfologicznej skutkuje załatwaniem „dziur” i wklęsłości na obrazie.

```
bim = imclose(bim,ones(5));
```



Rys. 25: Operacja zamknięcia.

2.11 Operacja otwarcia

Dokonanie następujących po sobie kolejno operacji erozji i dylatacji nazywamy operacją otwarcia. Zastosowanie tego algorytmu pozwala na wygładzenie krawędzi na obrazie przy jednoczesnym zachowaniu rozmiarów wygładzanego obiektu.

```
bim = imopen(bim,ones(5));
```



Rys. 26: Operacja otwarcia.

2.12 Wnioski

Poznane w trakcie tych ćwiczeń operacje pozwalają nam na odnajdywanie konturów obiektów znajdujących się na zdjęciu, a także na tworzenie masek dzięki którym jesteśmy w stanie wyłuskać interesujące nas obiekty oddzielając je od tła obrazu.

```
imshow(im.*bim); % wykorzystanie uprzednio przygotowanej maski
```



Rys. 27: Oryginalny obraz z nałożoną maską.

3. Laboratorium 4

W kolejnych zajęciach laboratoryjnych zapoznaliśmy się z transformatą Fouriera. Działanie poszczególnych testowanych funkcji sprawdzaliśmy na poniższym obrazie:



Rys. 28: Oryginał obrazu wykorzystywanego podczas laboratorium 4.

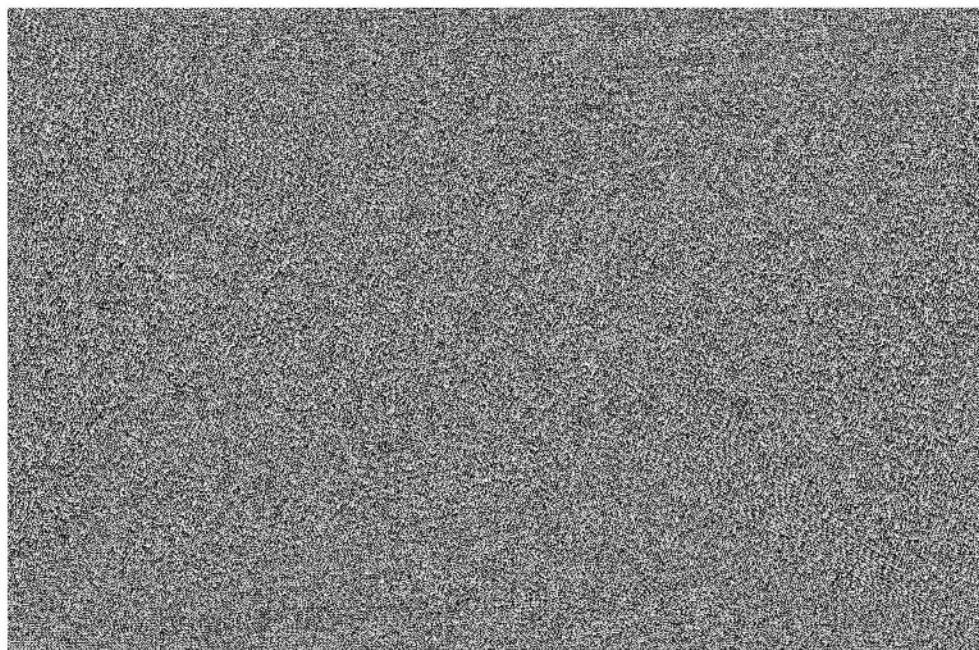


Rys. 29: Oryginał obrazu wykorzystywanego podczas laboratorium 4 w skali szarości.

3.1 Szybka transformata Fouriera

Korzystając z wbudowanej funkcji *fft2* w programie MATLAB dokonujemy na naszym obrazie szybkiej transformaty Fouriera. Wynikiem działania tej funkcji macierz liczb zespolonych o wymiarze równym wymiarowi wczytanego zdjęcia. Macierz wynikową możemy przekształcić do postaci zawierającej informacje o fazie i amplitudzie.

```
gim = rgb2gray(im);
f = fft2(gim);
amp = abs(f);
phase = angle(f);
imshow(phase,[-pi;pi]);      % faza powtarza się co 2*pi stąd
                                % zakres [-pi;pi]
```

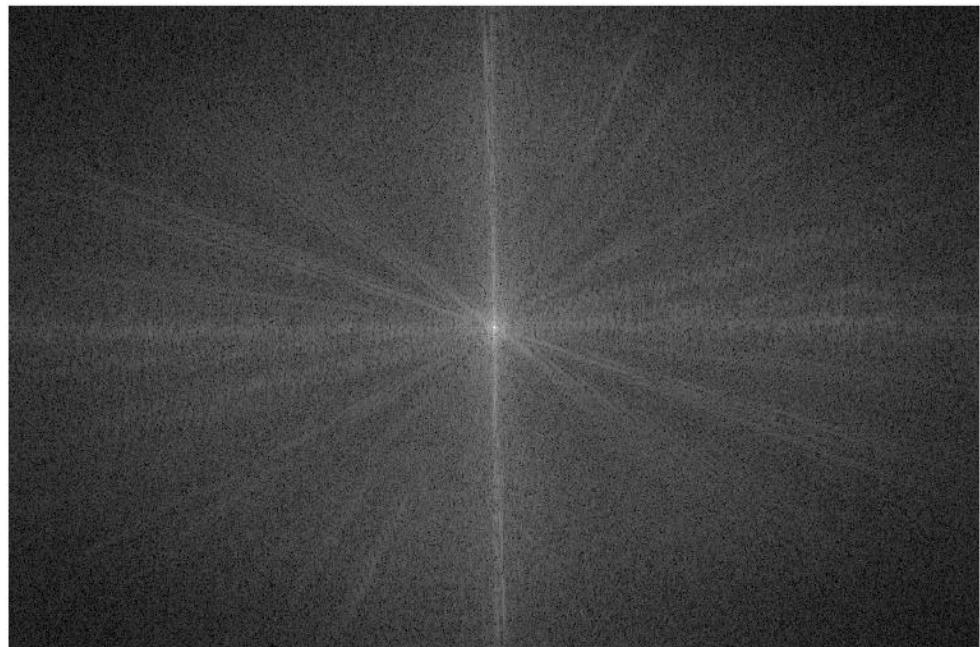


Rys. 30: Widmo fazowe oryginalnego obrazu (Rys.29).

Odnajdując regularne kształty w widmie fazowym obrazu jesteśmy w stanie stwierdzić, że analizowany przez nas obraz był uprzednio modyfikowany.

Kolejnym krokiem było poznanie pojęcia widma amplitudowego. Na jego podstawie jesteśmy w stanie określić w których częściach obrazu pojawiają się linie.

```
amp = abs(f);
maxamp = max(amp,[],'all');
imshow(fftshift(log(amp)),[0,log(maxamp)]);
```



Rys. 31: Widmo amplitudowe oryginalnego obrazu (Rys.29).

3.2 Transformata odwrotna i przywracanie obrazu do oryginalnej postaci

By dokonać odwrotnej transformaty Fouriera (funkcja `ifft2`) musimy przywrócić naszą amplitudę i fazę do postaci kanonicznej:

```
f = amp .* exp(1i*phase);      % zmiana na postać kanoniczną
im2 = abs(ifft2(f));          % transformata odwrotna
imshow(im2);
```

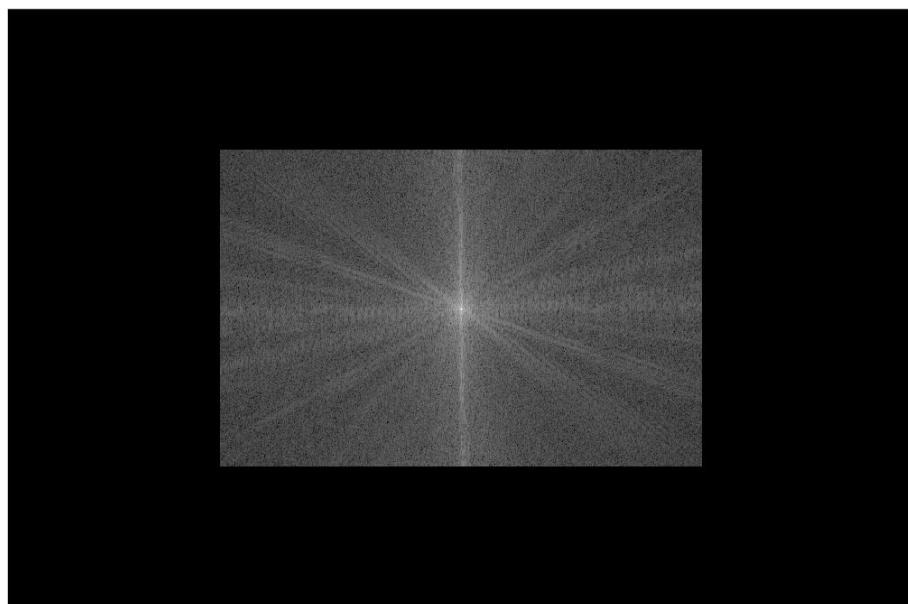


Rys. 32: Przywrócony obraz do pierwotnej wersji z wykorzystaniem odwrotnej transformaty Fouriera.

3.3 Kompresja

Zanim jednak dokonamy powrotu do oryginalnej postaci obrazka z wykorzystaniem odwrotnej transformaty, możliwa jest manipulacja amplitudy, dzięki której uzyskamy skompresowany obraz. Wykorzystując informację odczytaną z widma amplitudowego – najwięcej jasnych linii znajduje się w jej środku, a więc w tej części znajduje się najwięcej interesujących nas informacji znajdujących się na obrazie – możemy wykorzystać jedynie centralną część tego widma do tworzenia postaci kanonicznej.

```
maxamp = max(amp,[],'all');
[h,w] = size(gim);
mask = zeros(h,w);
mask(200:h-200, 300:w-300) = 1;      % ograniczamy interesujący nas
                                         % obszar widma amplitudowego
                                         % (ramki góra-dół = 200px;
                                         % lewo-prawo = 300px)
amp = amp.*fftshift(mask);           % nałożenie maski na widmo amp.
imshow(fftshift(log(amp)),[0,log(maxamp)]);
```



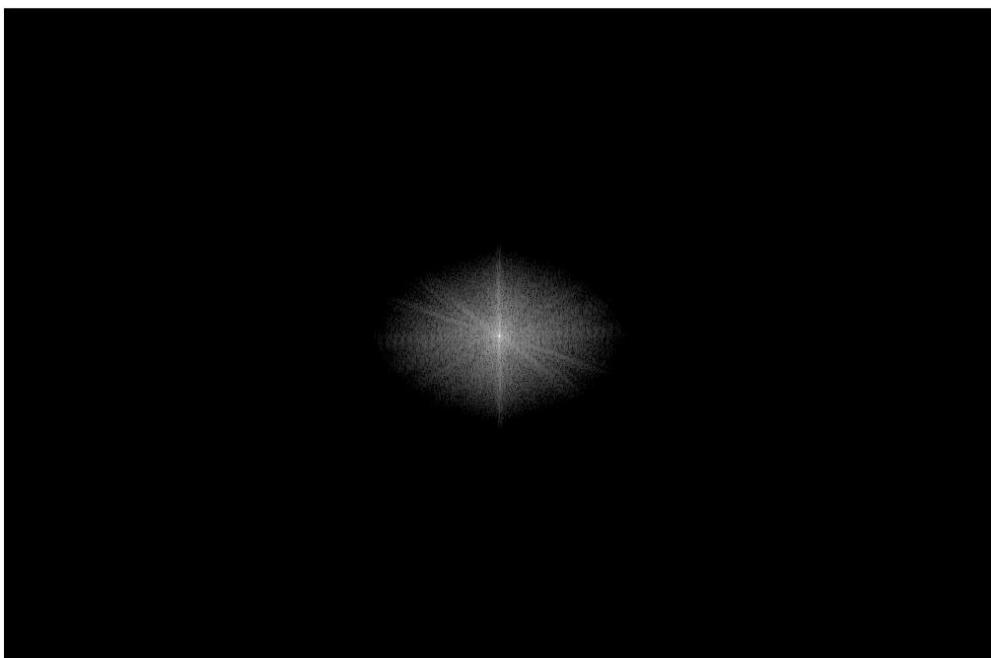
Rys. 32: Widmo amplitudowe z nałożoną maską.

Tak utworzoną maskę możemy wykorzystać do przywrócenia obrazu do postaci oryginalnej, pomniejszonego o informacje zawarte w pomijanej części widma amplitudowego (powtarzając analogicznie instrukcje zawarte w podpunkcie 3.2).



Rys. 33: Obraz po kompresji.

Kompresji z wykorzystaniem informacji zawartych w widmie amplitudowym dokonaliśmy również wykorzystując maskę opartą o filtr dolnoprzepustowy Gaussa.



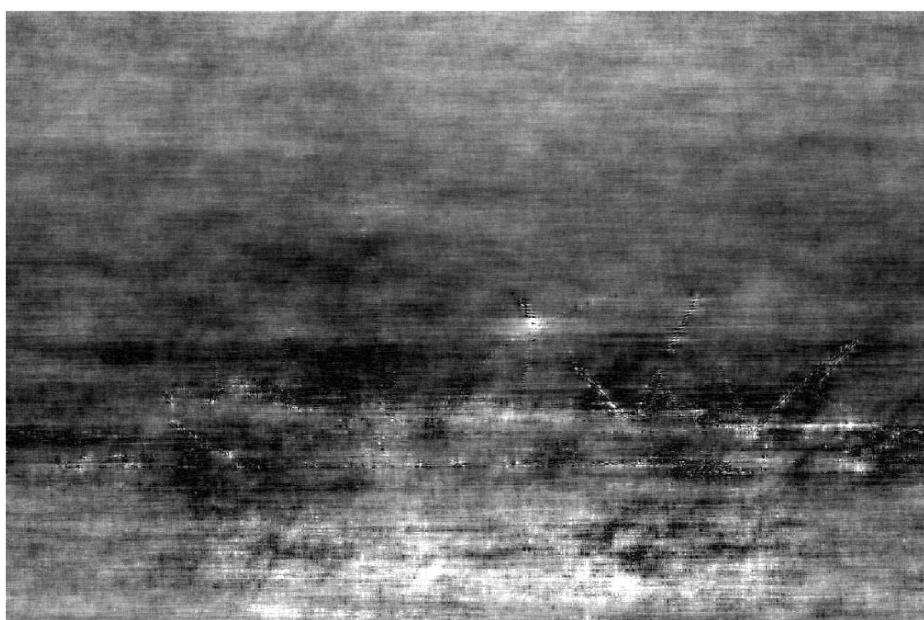
Rys. 34: Widmo amplitudowe z nałożonym filtrem dolnoprzepustowym Gaussa.



Rys. 35: Efekt przywrócenia obrazu po nałożeniu na widmo amplitudowe filtru dolnoprzepustowego Gaussa.

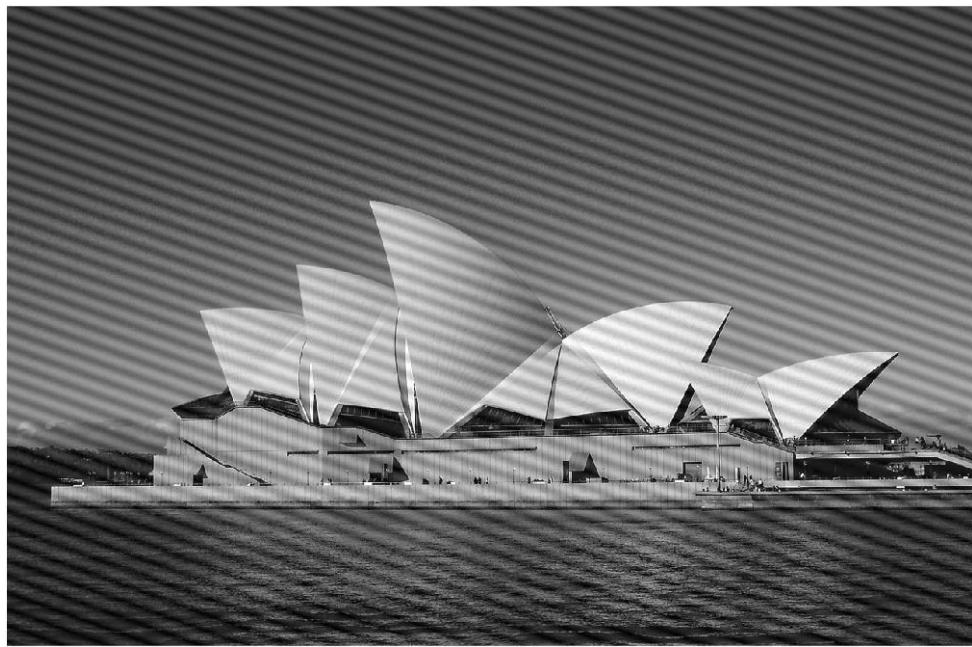
3.4 Zaburzenia obrazu

Faza zawiera istotne informacje o obrazie, stąd jej duża wrażliwość na zaburzenia. Użycie funkcji *fftshift* na widmie fazowym zniekształca znaczco obraz, czyniąc go nieczytelnym i trudnym do zinterpretowania. Poniżej się rezultat takiej ingerencji w widmo fazowe i próbie przywrócenia obrazu do oryginalnej postaci.



Rys. 36: Efekt ingerencji w widmo fazowe.

Ingerencja w widmo amplitudowe już na jednej jej wartości powoduje nadania „tekstury” na obrazie. Efekt, choć znaczący, to jednak w przeciwieństwie do ingerencji w widmo fazowe, pozwala na odczytanie zawartości obrazu po jego przywróceniu.



Rys. 37: Efekt ingerencji w widmo amplitudowe.

3.5 Wnioski

Szybka transformata Fouriera pozwala nam na uzyskanie skompresowanego obrazu bez utraty dużej ilości informacji. Widmo fazowe jest wrażliwe na wszelakie zniekształcenia i pozwala nam na zweryfikowanie autentyczności obrazu (tzn. czy obraz nie był uprzednio manipulowany).