**Laboratorium 1**

**Zadanie 1. Konwersja obrazu na różne formaty przy użyciu MATLAB i Image Processing Toolbox**

Użyj funkcji dostępnych w bibliotece IPT w MATLAB-ie do przekształcenia obrazu. Na Rys. 9 przedstawiono schemat przepływu i konwersji różnych typów obrazów w MATLAB-ie, z wykorzystaniem funkcji dostępnych w IPT.

*Rys. 9. schemat przepływu i konwersji różnych typów obrazów w MATLAB-ie.*

W zadaniu:

* Wczytaj dowolny obraz kolorowy (RGB) za pomocą funkcji imread.
* Dokonać konwersji na obraz monochromatyczny.
* Dokonać binaryzacji obrazu.
* Przekształcić obraz do formatu indeksowanego.
* Zwizualizować osobno kanały obrazu RGB.
* Zapisać wszystkie przekształcone obrazy do plików (imwrite).

Ponadto odpowiedź na poniższe pytania:

* Jak zmienia się zakres wartości pikseli w kolejnych przekształceniach?
* Jaka jest różnica w zajmowanej pamięci między różnymi formatami obrazów?
* Jakie mogą być zastosowania obrazów indeksowanych i binarnych w praktyce?

**Zadanie 2. Histogram i operacje na histogramie**

W Matlab zaproponuj program, który pozwoli na:

* Wczytanie obrazu w skali szarości (imread → rgb2gray jeśli potrzeba).
* Obliczenie i wyświetlenie histogramu obrazu (imhist).
* Wykonaj rozciąganie histogramu (imadjust + stretchlim) i porównaj obraz przed i po operacji.
* Wykonaj wyrównanie histogramu (histeq) i porównaj wynik z oryginalnym obrazem oraz z histogramem rozciągniętym.
* Wyświetl histogramy wszystkich przekształconych obrazów.

Pytania do analizy:

* Jaka jest różnica między rozciąganiem a wyrównywaniem histogramu?
* Która metoda daje lepsze wyniki poprawy kontrastu?
* Jakie są potencjalne zastosowania tych technik?

**Zadanie 3. Segmentacja obrazu na podstawie histogramu**

W Matlab zaproponuj program, który pozwoli na:

* Wczytanie obrazu w skali szarości.
* Obliczenie histogramu obrazu (imhist) i znalezienie potencjalnego progu segmentacji.
* Przeprowadź ręczną segmentację obrazu, stosując różne wartości progowe (BW = I > threshold).
* Zastosuj automatyczną segmentację metodą Otsu (graythresh + imbinarize).
* Porównaj efekty różnych progowań, wyświetlając wyniki obok siebie.

Pytania do analizy:

* Jak dobrać optymalny próg segmentacji dla różnych obrazów?
* Kiedy metoda Otsu jest skuteczniejsza niż segmentacja ręczna?
* Jakie mogą być praktyczne zastosowania binaryzacji w analizie obrazów?

**Zadanie 4. Detekcja obiektów na obrazie binarnym**

W Matlab zaproponuj program, w którym:

* Wykonasz segmentację obrazu metodą Otsu i uzyskaj obraz binarny.
* Zidentyfikuj obiekty na obrazie binarnym za pomocą funkcji bwlabel lub regionprops.
* Wyznacz podstawowe właściwości wykrytych obiektów (np. powierzchnię, obwód).
* Narysuj prostokątne obramowania wokół wykrytych obiektów (rectangle).
* Zwizualizuj oryginalny obraz z nałożonymi obramowaniami.

Pytania do analizy:

* Jakie problemy mogą wystąpić przy detekcji obiektów w zależności od jakości segmentacji?
* Jak można poprawić dokładność detekcji?
* Jakie są praktyczne zastosowania tej metody (np. w biomedycynie, kontroli jakości)?

**Zadanie 5. Segmentacja wieloprogowa**

W Matlab zaproponuj program, w którym:

* Wczytasz obraz w skali szarości.
* Obliczysz histogram obrazu i znajdźiesz dwa optymalne progi metodą Otsu (multithresh).
* Utwórz obraz segmentowany na trzy klasy (imquantize).
* Zwizualizuj wynik segmentacji z użyciem różnych kolorów (label2rgb).
* Porównaj segmentację wieloprogową z binaryzacją jednokryterialną.

Pytania do analizy:

* W jakich przypadkach segmentacja wieloprogowa daje lepsze wyniki niż binaryzacja?
* Jakie są jej ograniczenia?
* Jak można poprawić jakość segmentacji?

Labolatorium 2

3. Zadania do samodzielnego rozwiązania

Zadanie 1.

 Wykorzystując dowolny obraz biometryczny tęczówki oka wykonaj program dokonujący

filtracji splotowej filtrami Robertsa, Prewitta, Sobela oraz Laplace’a. Uwzględnij w programie

możliwość zadawania rozmiaru maski kernela 3x3 oraz 5x5.

 Wyświetl wyniki filtracji w oknie wykorzystując funkcję subplot.

 Porównaj efekty filtracji dla różnych filtrów. Sprawdź wpływ zmiany rozmiaru maski filtra na

wynik końcowy.

 Wprowadź do obrazu zadaną porcję szumu Salt&Pepper wykorzystując funkcję imnoise.

 Przeprowadź na obrazie wejściowym filtrację medianową i uśredniającą a następnie porównaj

otrzymane wyniki. Ponów procedurę wykrycia krawędzi w obrazie; przedstaw w sprawozdaniu

dyskusję otrzymanych wyników.

 Wczytaj obraz kolorowy (peppers.png) i zastosuj filtrację na każdej składowej R, G, B osobno.

Połącz przefiltrowane składowe w nowy obraz i wyświetl wynik.

 Sprawdź, jak działa imfilter(I, h, 'same') w porównaniu do conv2(). Czy wyniki są identyczne?

Jakie są różnice?

Implementacja masek:

Labolatorium 3

Zadanie 1.

1. Załaduj obraz testowy oraz przygotuj obraz do realizacji dalszych ćwiczeń o rozmiarach:

[x,map]= imread('thintest.bmp');

2. Wykonaj program realizujący algorytm ścieniania, którego zestaw kerneli zamieszczono poniżej:

NORTH SOUTH WEST EAST

0 0 0 1 1 X 0 X 1 X X 0

X 1 X X 1 X 0 1 1 1 1 0

X 1 1 0 0 0 0 X X 1 X 0

X 0 0 X 1 X 0 0 X X 1 X

1 1 0 0 1 1 0 1 1 1 1 0

X 1 X 0 0 X X 1 X X 0 0

3. Do zaprogramowania kolejnych szablonów i masek wykorzystaj poniższy fragment kodu źródłowego

(funkcja intbin8 formuje 8-bitową reprezentację liczby dziesiętnej przekazywanej parametrem i):

%wartosci prawidlowe

s=[192 80 12 5 3 65 48 20];

m=[206 87 236 117 59 93 179 213];

[f,map] = imread('thintest.bmp');

S=[];

M=[];

for i=1:8

S(:,i)=intbin8(s(i));

M(:,i)=intbin8(m(i));

end

4. Sprawdź poprawność działania algorytmu na innym obrazie testowym, dla zmodyfikowanych

parametrów filtracji morfologicznej jak poniżej:

s=[192 2 12 5 3 65 48 20];

m=[206 58 236 117 59 93 179 213];

5. Wyświetl wyniki otrzymanych filtracji wraz z podpisem w oknie wykorzystując funkcję subplot.

6. Wyjaśnij zjawisko zmieniającej się szybkości wykonywania algorytmu w zależności od fazy jego

wykonywania.

7. Wykonaj wersję algorytmu realizującego szkieletyzację wg Chin-Wan-Stover-Iverson, przyjmując

zestaw kerneli jak poniżej

8. Przeprowadź dyskusję otrzymanych wyników jak dla pkt 5-6.

Wskazówka! Skrypt ułatwiający wykonanie części symulacyjnej ćwiczenia:

%Algorytm realizujący thining obrazów binarnych

%====================================================================

% VAR=1-metoda Arcelli-Cordella-Levialdi,

% VAR=2-metoda Chin-Wan-Stover-Iverson,

% S1 szablon dla metody I, M1- maska bitów dowolnych,

% S2/S3 szablon dla metody II, M2/M3- maska bitów dowolnych,

VAR= 2; %Wariant metody

s1= [192, 80, 12, 5, 3, 65, 48, 20];

s2= [81, 69, 21, 84, 80, 65, 5, 20];

s3= [1, 64];

m1= [206, 87, 236, 117, 59, 93, 179, 213];

m2= [95, 125, 245, 215, 87, 93, 117, 213];

m3= [17, 68];

%Zamiana na postac binarna

S1=[]; S2=[]; S3=[];

M1=[]; M2=[]; M3=[];

for i=1:8

S1(:,i)= intbin8(s1(i));

S2(:,i)= intbin8(s2(i));

M1(:,i)= intbin8(m1(i));

M2(:,i)= intbin8(m2(i));

end

for i=1:2

S3(:,i)= intbin8(s3(i));

M3(:,i)= intbin8(m3(i));

end

[f,map]=imread('ZG.bmp');

[nr, nc]= size(f);

g=f; %negatyw (f~=1);

%h=zeros(nr,nc); % Pojemnik na obraz przejsciowy

start= tic();

c=1; %Indeks do szablonu north, south, west, east

W= 0; %Zmodyfikowal chociaz jeden piksel 1 jesli nie 0

W4=0; %Licznik podcykli wdg ktorego przetwarzany jest obraz w okreslonym kierunku

l\_iteracji=0 %Liczba wykonanych iteracji na obrazie do zbiegniecia

z=0;

h=g;

while W4<5

disp(sprintf('Iteracja= %i,',l\_iteracji+1))

W=0;

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for i=2:nr-1

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for j=2:nc-1

%disp(sprintf('z=%i, Iteracja=%i, Wiersz= %i, Kolumna= %i, W4= %i, W=

%i,',c,l\_iteracji,i,j,W4,W))

if g(i,j)>0

molekula= [g(i,j+1); g(i-1,j+1); g(i-1,j); g(i-1,j-1); g(i,j-1); g(i+1,j-1); g(i+1,j);

g(i+1,j+1)];

if VAR==1 % metoda Arcelli

for z=c:c+1 %Przykladaj poszczegolne szablony

if (molekula.\*M1(:,z))==S1(:,z)

h(i,j)=0; W=1; break;

end

end

%==============================================

else %VAR==2 metoda Chin-Wan-Stover-Iverson

if (molekula.\*M2(:,8))==S2(:,8)

h(i,j)=0; W=1; %zmienil piksel

else

if (((((molekula.\*M3(:,1))==S3(:,1)) & (g(i,j+2)==0)) | ...

(((molekula.\*M3(:,2))==S3(:,2)) & (g(i+2,j)==0))))

%nic nie rób

else

for z=1:7

if (molekula.\*M2(:,z))==S2(:,z)

h(i,j)=0; W=1; %zmieni piksel

break;

end

end

end

end

%=====================================================

end

end %End od if g(i,j)>0

end %Koniec pierwszego for

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

end %Koniec drugiego for

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

l\_iteracji= l\_iteracji+1;

g=h;

%Jesli metoda pierwsza to kontroluj numer cyklu

if W==0

W4=W4+1;

else

W4=0;

end

if (VAR==1)&(W4==4)

break;

end

c=1+2\*rem(l\_iteracji,4); %Zmienia numer indeksu w tablicy filtrow

g= h;

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

end %Koniec While

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

toc(start)

imagesc((uint8(f)+2\*uint8(g))/max(max(uint8(f)+uint8(g))));

colormap(jet);

Labolatoium 04

Zadanie 1. Wykrywanie i normalizacja tęczówki oka przy użyciu transformacji Hougha

Zaproponuj implementację algorytmu wykrywania tęczówki w obrazie oka przy wykorzystaniu

transformacji Hougha w wersji kołowej. Następnie należy przeprowadzić normalizację wykrytej

tęczówki poprzez przekształcenie jej do współrzędnych biegunowych.

W tym celu należy:

1. Wczytać obraz oka i jeżeli jest to konieczne zmienić go do skali szarości tzw. gray level.

2. Przeprowadzić wykrywanie krawędzi: w tym celu zastosuj operator Canny'ego do wykrycia

krawędzi. Dostosuj próg detekcji krawędzi, aby uzyskać możliwie wyraźne kontury.

3. Wykonać wykrywanie okręgów metodą Hougha: w tym celu wykorzystaj funkcję

imfindcircles do wykrycia granicy między twardówką a tęczówką (większy okrąg), granicy

między tęczówką a źrenicą (mniejszy okrąg). Ustaw odpowiednie zakresy promieni dla obu

okręgów oraz zwizualizuj wykryte okręgi na obrazie.

4. Przeprowadzić transformację wyznaczonego pola tęczówki do postaci macierzowej: w tym celu

przeprowadź przekształcenie wykrytej tęczówki do współrzędnych biegunowych. Ustaw

odpowiednie parametry dla tej transformacji (promień podziału, zakres kąta). Zwizualizuj

przekształcony obraz w nowych współrzędnych biegunowych.

5. Dokonać wizualizacji wyników: w tym celu wyświetl wykryte okręgi na oryginalnym obrazie oraz

wyświetl obraz tęczówki w układzie biegunowym.

Przykładowe wyniki, które mogą być uzyskane w trakcie prowadzonych symulacji:

Rys. 6. Przykładowe wyniki: wyodrębnione pole tęczówki w obrazie oraz jego rozwinięta postać we

współrzędnych biegunowych

Poniżej zamieszczono przykład kodu, który można wykorzystać w celu zaimplementowania algorytmu

wykrywania tęczówki w obrazie oka przy wykorzystaniu transformacji Hougha w wersji kołowej.

%Program demonstrujący wykorzystanie wersji kołowej transformaty Hougha

%w zadaniu wyodrębniania pola tęczówki

%==========================================================================

clear all, close all, clc

oko = imread('oko2.jpg'); %lub oko1

oko= mat2gray(oko);

imshow(oko)

% Wszystkie wystąpienia ???? należy uzupełnić samodzielnie dobranymi

% parametrami transformaty

Rmin = ????; %twardówka - tęczówka

Rmax = ????;

[srodki\_j, promienie\_j] = imfindcircles(oko,[Rmin Rmax],'ObjectPolarity',??????, ...

'Sensitivity', ?????);

Rmin = ????; %źrenica - tęczówka

Rmax = ????;

[srodki\_c, promienie\_c] = imfindcircles(im2bw(oko, 0.12),[Rmin Rmax],'ObjectPolarity',...

'dark','Method', ??????, 'Sensitivity', ??????);

viscircles(srodki\_j, promienie\_j,'Color','b'); %Wizualizacja okręgów

viscircles(srodki\_c, promienie\_c,'LineStyle','--'); %Wizualizacja okręgów

hold on

title('Wykrywanie okręgów w obrazie');

hold off

% parametry normalizacji

promien\_rozdz = 90;

kat\_zakres = 360;

% Rozwijanie tęczówki do współrzędnych biegunowych

[polar\_tab, szum\_tab] = polar\_transform(oko, ...

srodki\_c(1), srodki\_c(2), promienie\_c, ...

srodki\_j(1), srodki\_j(2), promienie\_j, ...

promien\_rozdz, kat\_zakres);

figure

imagesc(polar\_tab)

colormap(gray)

Zadanie 2. Automatyczne zliczanie monet na obrazie z wykorzystaniem transformacji Hougha

Celem zadania jest automatyczne wykrycie, zliczenie oraz określenie nominałów monet na obrazie

wejściowym za pomocą transformacji Hougha dla okręgów. Należy także oszacować złożoność

obliczeniową metody oraz przetestować ją na różnych zbiorach obrazów. Zapoznaj się z poniższym

kodem i zaproponuj brakujące rozwiązania.

% Program pomocniczy demonstrujący w jaki sposób może zostać wykonane zliczanie monet z

% wykorzystaniem wersji kołowej transformaty Hougha

% ==========================================================================

clear all, close all, clc

% Załaduj obraz zawierający przykładowe okręgi -> plik dostępny w zasobach pakietu Matlab

A = imread('circlesBrightDark.png');

imshow(A)

Rmin = 15; % dobór tych parametrów ma kluczowe znaczenie dla

Rmax = 95; % dalszej pracy algorytmu

[srodki\_j, promienie\_j] = imfindcircles(A,[Rmin Rmax],'ObjectPolarity','bright');

[srodki\_c, promienie\_c] = imfindcircles(A,[Rmin Rmax],'ObjectPolarity','dark');

viscircles(srodki\_j, promienie\_j,'Color','b'); %Wizualizacja okręgów

viscircles(srodki\_c, promienie\_c,'LineStyle','--'); %Wizualizacja okręgów

hold on

title('Wykrywanie okręgów w obrazie');

hold off

% ==========================================================================

% Na podstawie kodu powyżej zliczyć monety w zbiorach monety1-4.jpg :)

% ==========================================================================

A = imread('monety4.jpg');

A= A(380:1262,711:1657);

A= imresize(A,0.5);

imshow(A>55)

A = edge(A, 'canny');

% dla jpg-ów może być potrzebna konwersja do gray

% A= (A(:,:,1)+A(:,:,2)+A(:,:,3))./3;

% lub A= mat2gray(A);

Rmin = 30; % Zakres promieni poszukiwanych okręgów

Rmax = 55;

[srodki, promienie] = imfindcircles(A,[Rmin Rmax], 'ObjectPolarity','bright');

if length(srodki)>0

viscircles(srodki, promienie,'Color','b'); %Wizualizacja okręgów

najw\_moneta= find(promienie==max(promienie)); % indeks największej monety

najmn\_moneta= find(promienie==min(promienie)); % indeks najmniejszej monety

viscircles(srodki(najw\_moneta,:), promienie(najw\_moneta) ,'Color','r'); %Wizualizacja

okręgów

viscircles(srodki(najmn\_moneta,:), promienie(najmn\_moneta) ,'Color','y'); %Wizualizacja

okręgów

hold on

title(['Zliczanie monet w obrazie. Liczba monet to: ' num2str(length(srodki))]);

hold off

end

% Mile widziany fragment zliczający sumę nominałów monet :)

% return

% ==========================================================================

% Przetwarzanie zbioru monety1.jpg

% Szacowanie złożoności obliczeniowej

% ==========================================================================

start= tic;

im = imread('monety1.jpg');

A=im;

A= (A(:,:,1)+A(:,:,2)+A(:,:,3))./3;

e = edge(A, 'canny', 0.6);

radii = 60:1:90; %15:1:40;

h = circle\_hough(e, radii, 'same', 'normalise');

peaks = circle\_houghpeaks(h, radii, 'nhoodxy', 11, 'nhoodr', 7, 'npeaks', 90);

imshow(im);

hold on;

for peak = peaks

[x, y] = circlepoints(peak(3));

plot(x+peak(1), y+peak(2), 'g-');

end

hold off

stop= toc(start);

laboratorium 5

Zadanie 1.

W oparciu o poniższy opis struktury algorytmu związanego z rozkładem w szereg Fouriera zaproponuj

jego implementację. Struktura algorytmu:

 Inicjalizacja – zamknięcie otwartych okien, czyszczenie pamięci, ustawienie parametrów.

 Generowanie funkcji wejściowej – tworzenie sygnału do analizy.

 Rozkład na harmoniczne – obliczanie współczynników Fouriera.

 Sumowanie składowych – rekonstrukcja funkcji.

 Wizualizacja wyników – rysowanie wykresów poszczególnych harmonicznych i sygnału

końcowego.

Poniżej przedstawiono schemat dla powyższej struktury algorytmu:

Rys. 3. Struktura algorytmu Transtormaty Fouriera

%% Fourier Series Decomposition

% ============================================

close all; clear all; clc;

% Basis description

roz = 5; % Number of harmonics

a = -180:1:180; % Angular measure in degrees

t = a \* pi / 180; % Convert to radians

Tmax = t(end) - t(1); % Period length

krok = t(2) - t(1); % Step size

% Define the different signal types

y = sin(t) + 0.1\*sin(20\*t); % Custom signal to analyze

y = (t > 0)\*2 - 1; % Try other signals for fun

% y = t;

y=sin(t)+0.2\*cos(5\*t)+0.1\*sin(10\*t);

% Plot the original signal

figure;

plot(t, y, 'g');

title('Original Signal');

grid on;

%% Fourier Series Coefficients

% ============================================

Ao = (1/Tmax) \* sum(y .\* krok);

A = zeros(roz, 1);

B = zeros(roz, 1);

for j = 1:roz

ac = 0; bs = 0;

for i = 1:length(t)

ac = ac + krok \* (2/Tmax) \* y(i) \* cos(j \* 2\*pi/Tmax \* t(i));

bs = bs + krok \* (2/Tmax) \* y(i) \* sin(j \* 2\*pi/Tmax \* t(i));

end

A(j) = ac;

B(j) = bs;

end

%% Reconstruct Signal from Harmonics

% ============================================

Skc = zeros(roz, length(t));

Sks = zeros(roz, length(t));

for i = 1:roz

Skc(i,:) = A(i) \* cos(i \* 2\*pi/Tmax \* t);

Sks(i,:) = B(i) \* sin(i \* 2\*pi/Tmax \* t);

end

% Initialize reconstructed signal with Ao/2

W = zeros(1, length(t)) + Ao / 2;

% Sum of harmonics

figure;

hold on;

for i = 1:roz

W = W + Skc(i,:) + Sks(i,:);

plot(t, Sks(i,:), 'b', t, Skc(i,:), 'g', t, W, 'r');

end

title('Harmonic Components and Signal Reconstruction');

xlabel('t');

ylabel('Amplitude');

grid on;

hold off;

%% Final Reconstructed Signal

figure;

plot(t, W, 'r', 'LineWidth', 1.5);

title('Final Reconstructed Signal from Fourier Series');

xlabel('t');

ylabel('Amplitude');

grid on;

%% Plot Coefficients

figure;

subplot(2,1,1);

% Stem plot or liner mode

% stem(1:roz, A, 'g', 'filled');

% Or linear mode

plot(1:roz,A, 'g')

title('Cosine Coefficients A\_n');

xlabel('Harmonic n');

ylabel('Amplitude');

grid on;

subplot(2,1,2);

stem(1:roz, B, 'r', 'filled');

title('Sine Coefficients B\_n');

xlabel('Harmonic n');

ylabel('Amplitude');

grid on;

%% Optional: Magnitude Spectrum

figure;

% stem(1:roz, sqrt(A.^2 + B.^2), 'k', 'filled');

plot(1:roz, sqrt(A.^2 + B.^2), 'r')

title('Fourier Magnitude Spectrum |C\_n|');

xlabel('Harmonic n');

ylabel('Amplitude');

grid on;

Zadanie 2.

Wykorzystując dwuwymiarową transformatę Fouriera zaproponuj jego implementację oraz wykonaj

eksperymenty z filtracją środkowo przepustową obrazu tęczówki oka podanej przez prowadzącego.

Poniżej przedstawiono przykładowe rezultaty uzyskane dla obrazu Lena z wykorzystaniem maski

prostokątnej.

(a) (b)

Rys. 4. Przykłady filtracji górnoprzepustowej (a) i dolnoprzepustowej (b)

% Step 1: Load and preprocess normalized iris image

close all; clear; clc;

% Load a normalized (polar) iris image

I = imread('teczowka\_2.png'); % Assumed grayscale, 64x512 or similar

if size(I,3) == 3

I = rgb2gray(I);

end

I = im2double(I);

% Display original

figure; imshow(I, []); title('Original Normalized Iris');

% Step 2: Compute 2D FFT and shift frequency spectrum

F = fft2(I);

F\_shifted = fftshift(F);

magF = log(1 + abs(F\_shifted));

figure; imshow(magF, []); title('Frequency Spectrum (log scale)');

% Step 3: Design bandpass filter

[rows, cols] = size(I);

[u, v] = meshgrid(-floor(cols/2):floor((cols-1)/2), -floor(rows/2):floor((rows-1)/2));

% The cone like mask shape

D = sqrt(u.^2 + v.^2);

% Define cutoff frequencies

D\_low = 5; % Remove very low frequency (lighting, iris boundary drift)

D\_high = 25; % Remove high freq noise (salt & pepper, reflections)

% Create ideal bandpass mask

H = double(D > D\_low & D < D\_high);

% Optional: visualize mask

figure; imshow(H, []); title('Bandpass Filter for Iris');

% Step 4: Apply filter

F\_filtered = F\_shifted .\* H;

% Step 5: Inverse FFT

I\_filtered = real(ifft2(ifftshift(F\_filtered)));

I\_filtered = mat2gray(I\_filtered);

% Step 6: Compare results

figure;

subplot(1,2,1); imshow(I, []); title('Original Normalized Iris');

subplot(1,2,2); imshow(I\_filtered, []); title('Enhanced Iris (Fourier Filtered)');

% Assesing space shape of the iris

figure

mesh(I\_filtered)

title('Morphology of the enrolled iris in the polar plane')

% Frequency fingerprint filtering

%===================================================

% Step 1: Load fingerprint image

close all; clear; clc;

I = imread('finger1.jpg'); % Use grayscale fingerprint

if size(I,3) == 3

I = rgb2gray(I);

end

I = im2double(I);

% Display original

figure; imshow(I); title('Original Fingerprint');

% Step 2: Compute 2D FFT and shift zero freq to center

F = fft2(I);

F\_shifted = fftshift(F);

magnitude = log(1 + abs(F\_shifted));

% Display frequency spectrum

figure; imshow(magnitude, []); title('Frequency Spectrum (log scale)');

% Step 3: Create bandpass filter mask

[rows, cols] = size(I);

[u, v] = meshgrid(-floor(cols/2):floor((cols-1)/2), -floor(rows/2):floor((rows-1)/2));

D = sqrt(u.^2 + v.^2);

% Define cutoff frequencies

D\_low = 10; % Suppress low frequencies (lighting, background)

D\_high = 60; % Suppress high freq noise

% Ideal bandpass filter

H = double(D > D\_low & D < D\_high);

% Optional: visualize filter

figure; imshow(H, []); title('Bandpass Filter Mask');

% Step 4: Apply filter in frequency domain

F\_filtered = F\_shifted .\* H;

% Step 5: Inverse FFT to get enhanced image

I\_filtered = real(ifft2(ifftshift(F\_filtered)));

% Normalize for display

I\_filtered = mat2gray(I\_filtered);

% Show enhanced result

figure;

subplot(1,2,1); imshow(I); title('Original');

subplot(1,2,2); imshow(I\_filtered); title('Enhanced via Fourier Filtering');

Zadanie 3.

Przetestuj przedstawiony powyżej mechanizm filtracji częstotliwościowej na obrazie

daktyloskopijnym.

% Frequency fingerprint filtering

%===================================================

% Step 1: Load fingerprint image

close all; clear; clc;

I = imread('finger1.jpg'); % Use grayscale fingerprint

if size(I,3) == 3

I = rgb2gray(I);

end

I = im2double(I);

% Display original

figure; imshow(I); title('Original Fingerprint');

% Step 2: Compute 2D FFT and shift zero freq to center

F = fft2(I);

F\_shifted = fftshift(F);

magnitude = log(1 + abs(F\_shifted));

% Display frequency spectrum

figure; imshow(magnitude, []); title('Frequency Spectrum (log scale)');

% Step 3: Create bandpass filter mask

[rows, cols] = size(I);

[u, v] = meshgrid(-floor(cols/2):floor((cols-1)/2), -floor(rows/2):floor((rows-1)/2));

D = sqrt(u.^2 + v.^2);

% Define cutoff frequencies

D\_low = 10; % Suppress low frequencies (lighting, background)

D\_high = 60; % Suppress high freq noise

% Ideal bandpass filter

H = double(D > D\_low & D < D\_high);

% Optional: visualize filter

figure; imshow(H, []); title('Bandpass Filter Mask');

% Step 4: Apply filter in frequency domain

F\_filtered = F\_shifted .\* H;

% Step 5: Inverse FFT to get enhanced image

I\_filtered = real(ifft2(ifftshift(F\_filtered)));

% Normalize for display

I\_filtered = mat2gray(I\_filtered);

% Show enhanced result

figure;

subplot(1,2,1); imshow(I); title('Original');

subplot(1,2,2); imshow(I\_filtered); title('Enhanced via Fourier Filtering

laboratorium 6

5. Zadania do samodzielnego rozwiązania

Zadanie 1.

Wykonaj tokenizację wybranej pary obrazów daktyloskopijnych i wyznacz podobieństwa zestawów

wyodrębnionych minucji z wykorzystaniem metody prostej oraz grafowej. Porównaj skuteczność obu

metod.

Zadanie 2.

Wykonaj tokenizację wybranej podanego zestawu obrazów tęczówki oka i wyznacz dla każdej pary

dystans Hamminga HD. Oceń otrzymane wyniki ze względu na podobieństwo i poziom okluzji pola

tęczówki.