KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Instrukcja współfinansowana przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego
w projekcie

*„Innowacyjna dydaktyka bez ograniczeń
– zintegrowany rozwój Politechniki Łódzkiej – zarządzanie Uczelnią,
nowoczesna oferta edukacyjna i wzmacniania zdolności
do zatrudniania osób niepełnosprawnych”*

Instrukcja jest dystrybuowana bezpłatnie.

Instrukcja do laboratorium, część 5-6

Marcin Kociołek

Metody przetwarzania i analizy obrazów biomedycznych

Zadanie nr 13 – Studia podyplomowe „Przetwarzanie i analiza obrazów biomedycznych”

Politechnika Łódzka
Instytut Elektroniki

90-924 Łódź, ul. Żeromskiego 116,
tel. 042 631 28 83
www.kapitalludzki.p.lodz.pl

Temat ćwiczenia:

Operacje na macierzach zawierających obrazy w środowisku MatLab

Cel ćwiczenia:

Głównym celem ćwiczenia jest zaprezentowanie iteracyjnego dostępu do pikseli obrazu w środowisku MatLab w zadaniach segmentacji obrazów.

Celem dodatkowym jest ugruntowanie wiedzy i umiejętności dotyczących sposobu obsługiwania obrazów rastrowych w systemach komputerowych.

[Zadanie 1] Operacje na wybranych wierszach i kolumnach obrazu.

Utworzyć plik piksele.m

Uwaga! Test w kolorze zielonym poprzedzony znakiem % jest komentarzem. Nie jest on konieczny do poprawnego działania skryptu. Należy jednak zaznaczyć, że umieszczanie komentarzy w plikach źródłowych lub w skryptach podnosi ich czytelność, pozwala w łatwiejszy sposób pracować nad danym zagadnieniem w grupie oraz ułatwia analizę programu po pewnym czasie.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Operacje wstępne
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

iptsetpref('ImshowBorder', 'tight');    %obrazy będą wyświetlane
                                         %z wąską ramką
clc;        % Czyszczenie okna poleceń.
clear;      % Kasowanie wszystkich zadeklarowanych zmiennych.
close all;  % Zamykanie wszystkich otwartych okien z obrazami (figure).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Utworzenie obrazu zawierającego zera
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

maxX = 200;     %Definicja szerokości obrazu.
maxY = 100;     %Definicja Wysokości obrazu.
Obr = zeros(maxY,maxX)  % Tworzenie obrazu (macierzy dwuwymiarowej)
                        % zawierającego zera. Jest to jednocześnie
                        % zarezerwowanie odpowiedniej ilości pamięci
                        % komputera co przyspieszy późniejsze operacje

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Wyświetlenie obrazu
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1);          % Otwarcie okna obrazu o numerze 1.
                    % Jeżeli takie okno istnieje, to ta funkcja
                    % uczyni okno numer 1 aktualnym.

imshow(Obr);        % Wyświetlenie obrazu
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Rysowanie poziomej kreski
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
pozycjaKreskiY = 50;


for y =1:maxY
    for x=1:maxX
        if y == pozycjaKreskiY
            Obr(y,x) = 1;
        end
    end
end

figure(2);
imshow(Obr);
```

Zmodyfikować ten skrypt w taki sposób, aby kreska była na brzegu obrazu.

Narysować kreskę poziomą.

Narysować kreskę skośna

Czy do rysowanie poziomej kreski jest potrzebne przeszukiwanie całego obrazu?

Zmodyfikować skrypt w tali sposób, aby wykorzystać pojedynczą pętlę for

Napisać skrypt rysujący ramkę w odległości 5 pikseli od brzegu obrazu.

[Zadanie 2] Wyświetlenie obrazu w skali szarości i w pseudokolorach.

Utworzyć plik segm.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Operacje wstępne
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

iptsetpref('ImshowBorder', 'tight');    %obrazy będą wyświetlane
                                         %z wąską ramką
clc;        % Czyszczenie okna poleceń.
clear;      % Kasowanie wszystkich zadeklarowanych zmiennych.
close all;  % Zamykanie wszystkich otwartych okien z obrazami (figure).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Otwarcie pliku
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Obr = imread('komórk1_1a.tif'); % odczyt obrazu i przypisanie go
                                % do zmiennej Obr
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           wyświetlenie obrazu w skali szarości
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1);              % Otwarcie okna obrazu o numerze 1.
                        % Jeżeli takie okno istnieje, to ta funkcja
                        % uczyni okno numer 1 aktualnym.

maxObr = max(max(Obr))  % Wyznaczenie maksymalnej jasności obrazu.
                        % Obraz jest macierzą dwuwymiarową, a więc
                        % w MatLabie należy funkcje max wywołać
                        % dwukrotnie. Pierwsze wywołanie tej funkcji
                        % utworzy wektor (macierz jednowymiarowa)
                        % zawierającą wartości maksymalne we wszystkich
                        % kolumnach. Drugie wywołanie tej funkcji da
                        % największa wartość w tym wektorze. Ta wartość
                        % jest jednocześnie maksymalna wartością w całym
                        % obrazie
minObr = min(min(Obr))  % Wyznaczenie maksymalnej jasności obrazu.

imshow(Obr,[minObr maxObr]);    % Wyświetlenie obrazu w zakresie jasności
                                % od minObr do maxObr

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Wyświetlenie obrazu w pseudokolorach
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Paleta barw pseudo kolorów o nazwie 'Jet' posiada 64 wartości
% Tworzony jest obraz tymczasowy ObrTemp, w którym wartości zostaną
% przeskalowane do zakresu 0 - 63

a = 63/double(maxObr - minObr);                 % Wyznaczenie
współczynników
b = 63*double(minObr)/double(maxObr - minObr);  % przeskalowania wart
pixeli

[maxY maxX] = size(Obr)     % Wyznaczenie rozmiarów obrazu Obr
ObrTemp = zeros(maxY,maxX); % Tworzenie obrazu pustego/deklaracja pamięci.

for y =1:maxY
    for x=1:maxX
        ObrTemp(y,x) = Obr(y,x) * a - b;    % Obliczanie wartości pikseli
                                            % obrazu tymczasowego.
    end
end

figure(2);      % uaktywnienie/otwarcie okna nr 2
imshow(ObrTemp,colormap('Jet'));    % Wyświetlanie obrazu tymczasowego
                                    % w palecie barw 'Jet'
```

[Zadanie 3] Progowanie ręczne.

Wyświetl histogram obrazu na uzupełniając skrypt segm.m o następujące polecenia.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Wyświetlenie histogramu
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(3);              % Uaktywnienie/otwarcie okna nr 3
imhist(Obr,2^16);       % Wyświetlenie histogramu
```

Korzystając z linii poleceń (Command Window) przeskaluj wykres tak, aby można było znaleźć wartość progu do segmentacji.

```
axis([100 500 0 200]);  % zmiana zakresu wyświetlanych wartości
    %  ^   ^  ^  ^- maksymalna wartość na skali  Y
    %  |   |  |- minimalna wartość na skali  Y
    %  |   |- maksymalna wartość na skali  X
    %  |- minimalna wartość na skali  X
```

Uzupełniamy skrypt segm.m o następujące linie podając w miejsce XX wyznaczona wartość.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Progowanie
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

prog = XX;                  % Wartość progu ustawiona ręcznie
ObrBw = zeros(maxY,maxX);   % Tworzenie obrazu pustego/deklaracja pamięci.
                            % Obraz Bw będzie zawierał wynik progowania

for y =1:maxY
    for x=1:maxX
        if Obr(y,x) < prog  % Jeżeli w obrazie Obr wartość piksela
                            % o współrzędnych y,x jest poniżej progu
            ObrBw(y,x) = 0; % to w obrazie ObrBw piksel o tych samych
                            % współrzędnych ustawiamy na zero
        else                % Jeżeli inaczej
            ObrBw(y,x) = 1; % to w obrazie ObrBw piksel o tych samych
                            % współrzędnych ustawiamy na 1
        end
    end
end

figure(4)
imshow(ObrBw);
```

[Zadanie 4] Dzielenie obrazu na segmenty.

Uzupełnij skrypt segm.m o następujące linie

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Wyznaczenie oddzielnych segmentów
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[ObrSegm,liczbaSegm] = bwlabel(ObrBw,4);    % Funkcja ta zwraca obraz
                                            % podzielony na segmenty
liczbaSegm

%Tworzenie obrazu tymczasowego do wyświetlenia segmentów w różnych
%kolorach
ObrTemp = label2rgb(ObrSegm, @spring, 'c', 'shuffle');
figure(5)
imshow(ObrTemp);
```

[Zadanie 5] Usuwanie segmentów o zbyt małych powierzchniach.

Uzupełnij skrypt segm.m o następujące linie

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Usuwanie segmentów o małych powierzchniach
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%liczymy pola powierzchni
%tworzymy wektor, który będzie zawierał pola powierzchni
PolaPowierzchni = zeros(liczbaSegm,1);

for y =1:maxY
    for x=1:maxX
        index = ObrSegm(y,x);
        if index > 0
            PolaPowierzchni(index) = PolaPowierzchni(index) + 1;
        end
    end
end
PolaPowierzchni
```

Zmodyfikuj fragment powyższego kodu, aby wyznaczyć pole powierzchni tła.

Uzupełnij skrypt segm.m o następujące linie.

```
% usuwanie obszarów o powierzchni mniejszej niż
minPowierzchnia = 400;

for y =1:maxY
    for x=1:maxX
        index = ObrSegm(y,x);
        if index > 0
            if PolaPowierzchni(index) <= minPowierzchnia;
                ObrSegm(y,x) = 0;
            end
```

```
        end
    end
end

ObrTemp = label2rgb(ObrSegm, @spring, 'c', 'shuffle');
figure(5)
imshow(ObrTemp);
```

Zmodyfikuj skrypt w taki sposób, aby usunąć obszary dotykające brzegu obrazu

[Zadanie 6] Przenumerowywanie obszarów.

Uzupełnij skrypt segm.m o następujące linie.

```
%przenumerowanie obszarów
for y =1:maxY
    for x=1:maxX
        index = ObrSegm(y,x);
        if index > 0
            ObrBw(y,x) = 1;
        else
            ObrBw(y,x) = 0;
        end
    end
end

[ObrSegm,liczbaSegm] = bwlabel(ObrBw,4);
liczbaSegm
```

Stwórz okno rysunkowe o numerze 6. Wyświetl obraz segmentów po przenumerowaniu. Użyj celu procedurę z zadania 4.

Co uległo zmianie? Dlaczego przenumerowanie jest istotne?

[Zadanie 7] Wyznaczanie cech obszarów (średnice Fereta).

Uzupełnij skrypt segm.m o następujące linie.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cechy obiektów
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%deklaracje wektorów zawierających liczby do wyznaczenia cech obszarów
%wszystkie deklarowane wektory(macierze jednowymiarowe) mają rozmiar
%odpowiadający ilości segmentów w obrazie
SegPolePow = zeros(liczbaSegm,1);       %pola powierzchni
SegMinX = ones(liczbaSegm,1)*maxX;      %minimalna wartość współrzędnej X
```

```
%Pętla do wyznaczania cech w poszcególnych obszarach
for y =1:maxY
    for x=1:maxX
        index = ObrSegm(y,x);
        if index > 0            %obliczenie przeprowadzamy gdy analizowany
                                %piksel nie nalerzy do tła
            SegPolePow(index) = SegPolePow(index) + 1;  %inkrementacja
                                                        %(zwiększanie o jeden)
            if SegMinX(index) > x
                SegMinX(index) = x;
            end
            if SegMaxX(index) < x
                SegMaxX(index) = x;
            end
        end
    end
end

% wyświetlanie wartości
SegPolePow
SegMinX
SegMaxX
SegFeretX = SegMaxX - SegMinX   %średnica Fereta w kierunku x
```

Zmodyfikuj skrypt w taki sposób, aby wyznaczyć wartości SegFeretX.

[Zadanie 8] Wyznaczanie cech obszarów (średnia jasność).

Uzupełnij fragment kodu z zadania 7 tak, aby wyznaczyć sumę jasności pieli w poszczególnych obszarach.

Do deklaracji wektorów cech dodaj nową zmienna

```
SegSumaJasnosc = double(zeros(liczbaSegm,1));   %suma jasności pikseli
                                                %w obrazie oryginalnym
```

Do pętli wyznaczającej cechy dopisz funkcję:

```
SegSumaJasnosc(index) = SegSumaJasnosc(index) + double(Obr(y,x));
                    %zwiększanie sumy o wartość jasności aktualnego piksela
```

Do wyświetlania wartości dopisz:

```
SegSredJasnosc = SegSumaJasnosc ./ SegPolePow   %średnia jasność pikseli to
                                                %suma wartości ich jasności
                                                %dzielona przez
                                                %liczbę pikseli
%operator ./ oznacza dzielenie tablicowe. Każdy element wektora
%SegSumaJasnosc zostanie podzielony przez przed odpowiadający mu element
%wektora SegPolePow
```

[Zadanie 9] Wyznaczanie konturu obszaru (sąsiedztwo 4).

Uzupełnij skrypt segm.m o następujące linie.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Wyznaczenie konturu obrszaru
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ObrSegmKontur = zeros(maxY,maxX);
for y =2:(maxY-1)
    for x=2:(maxX-1)
        index = ObrSegm(y,x);   %analizowany pixel
        if index > 0
            indexO1 = ObrSegm(y-1,x);   %Sąsiad analizowanego piksela
            indexO2 = ObrSegm(y+1,x);
            indexO3 = ObrSegm(y,x-1);
            indexO4 = ObrSegm(y,x+1);
            if indexO1 == index & indexO2 == index & indexO3 == index ...
                    & indexO4 == index
                                % ... oznacza że wyrażenie z bieżącej
                                %linii będzie kontynuowane niżej

                ObrSegmKontur(y,x) = 0;
            else
                ObrSegmKontur(y,x) = index;
            end
        end
    end
end

ObrTemp = label2rgb(ObrSegmKontur, @spring, 'c', 'shuffle');
figure(7)
imshow(ObrTemp);
```

Uzupełnij skrypt segm.m o funkcję obliczającą liczbę pikseli konturów.

[Zadanie 10] Wyznaczanie konturu obszaru (sąsiedztwo 8).

Uzupełnij skrypt segm.m o funkcje wykreślającą kontur w sąsiedztwie 8.

Uzupełnij skrypt segm.m o funkcję obliczającą liczbę pikseli konturów.

Czym różnią się te dwa kontury?

[Zadanie 11] Tworzenie obrazu zawierającego wybrany pojedynczy obiekt.

Uzupełnij skrypt segm.m o następujące linie.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% tworzenie obrazu zawierającego wybrany pojedynczy seegment
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
indexSegm = 2;

maxSegY = SegMaxY(indexSegm) - SegMinY(indexSegm)
maxSegX = SegMaxX(indexSegm) - SegMinX(indexSegm)

ObrPojedynczegoSegmentu = zeros(maxSegY,maxSegY);

offsetSegY = SegMinY(indexSegm);
offsetSegX = SegMinX(indexSegm);

for y =1:maxSegY
    for x=1:maxSegX
        index = ObrSegm(y+offsetSegY,x+offsetSegX);
        if index == indexSegm;
            ObrPojedynczegoSegmentu(y,x) = Obr(y+offsetSegY,x+offsetSegX);
        end
    end
end


figure(8)
ObrTemp = double(ObrPojedynczegoSegmentu) *a - b;
imshow(ObrTemp,colormap('Jet'));
```