Przetwarzanie obrazów cyfrowych w środowisku MATLAB

Marcin Iwanowski

2 listopada 2017

Spis treści

1 Wprowadzenie

Proces przetwarzania obrazu i rozpoznawania jego treści składa się zwykle z kilku etapów. Można w nim wyróżnić następujące kroki:

1. **Akwizycja** polega na przetworzeniu fragmentu rzeczywistości obserwowanego przez sensor wizyjny (np. kamerę) na postać cyfrową w formie macierzy punktów obrazu - pikseli.
2. **Filtracja** ma na celu usunięcie z obrazu zbędnych elementów, przede wszystkim szumu i/lub poprawę jakości obrazu pod kątem dalszej jego obróbki.
3. **Segmentacja** polega na wyodrębnieniu na obrazie obiektów na nim się znajdujących i zapamiętaniu ich w odrębnym obrazie.
4. **Wyznaczanie cech obrazu**, dzięki któremu każdy z wyodrębnionych obiektów jest opisywany przy pomocy specjalnych parametrów. Cechy opiują właściwości oniektów istotne z punktu widzenia ich identyfikacji.
5. **Klasyfikacja**, w trakcie której obiekty są klasyfikowane na podstawie swoich cech do odpowiednich klas. Na etapie rozpoznawania możliwa jest także ocena jakościowa analizowanych obiektów.

W trakcie ćwiczeń pokazane zostaną najczęściej stosowane operacje, wykorzystywane na każdym z kolejnych etapów (z pominięciem etapu akwizycji) oraz przedstawione zadania do samodzielnego wykonania. Ćwiczenia mają na celu pokazanie podstawowych kroków projektowania systemu wizyjnego. Głównym zadaniem projektanta takiego systemu jest bowiem odpowiedni dobór kolejno wykonywanych operacji przetwarzania obrazów, a także ich optymalnych parametrów pod kątem konkretnego zadania wykonywanego przez system.

2 Przetwarzanie obrazów w środowisku MATLAB

MATLAB jest uniwersalnym środowiskiem obliczeniowym. Pierwotnie był pakietem służącym do obliczeń macierzowych. Dzięki licznym rozszerzeniom (ang. toolboxes) stał się narzędziem do wszechstronnych obliczeń naukowych i technicznych. Rozszerzenie Image Processing Toolbox pozwala na wykonywanie operacji przetwarzania i rozpoznawania obrazów. Z tego powodu jest doskonałym narzędziem do wykonywania operacji odpowiadającym kolejnym etapom przetwarzania typowych dla systemów wizyjnych.

2.1 Rodzaje obrazów

Obrazy w MATLAB-ie reprezentowane są jako macierze. Pojedynczy element obrazu - piksel odpowiada więc elementowi macierzy. Liczba wymiarów macierzy odpowiada liczbie wymiarów obrazu. Tak więc obrazy dwuwymiarowe wykorzystywane w ćwiczeniu zapisywane są jako dwuwymiarowe macierze prostokątne. Macierz obrazu zawiera dane określonego typu odpowiadającego jednemu ze standardowych typów danych zdefiniowanych w środowisku MATLAB-a. Od typu danych wykorzystywanych do reprezentacji obrazu zależy wielkość obszaru zajmowanego przez obraz w pamięci komputera oraz szybkość jego przetwarzania. W praktycznych zastosowaniach korzysta się z następujących typów danych MATLAB-a:

`double`	Podwójna precyzja - liczba zajmuje 8 bajtów pamięci, zakres do -10^{308} do 10^{308}, w przetwarzaniu obrazów wykorzystuje się liczby z zakresu $< 0, 1 >$.
`uint8`	8-bitowa liczba całkowita bez znaku
`uint16`	16-bitowa liczba całkowita bez znaku

MATLAB pozwala na pracę z następującymi czterema rodzajami obrazów.

Obrazy binarne Piksel obrazu binarnego może przybierać wartości 0 lub 1. Obrazy takie wykorzystują typy danych `double` lub `uint8`.

Obrazy szarościowe W obrazie w skali szarościm piksel może przybierać wartości skalarne z zadanego zakresu. W przypadku obrazów typu `double` wartości są liczbami podwójnej precyzji z zakresu $< 0, 1 >$. W obrazach `uint8` wartść punktu jest liczbą naturalną nie większą niż 255, w przypadku zaś obrazów `uint16` maksymalna wartość punktu wynosi 65535.

Obrazy kolorowe nieindeksowane Wartość piksela obrazu kolorowego jest wielkością wektorową, będącą najczęściej trójką liczb opisujących wartość w paśmie czerwonym, zielonym i niebieskim (dla modelu barw RGB). Obraz taki składa się więc z trzech warstw odpowiadających poszczególnym składowym koloru: R,G i B. Każda warstwa jest obrazem skalarnym własnościach identycznych z obrazem w skali szarościm.

Obrazy kolorowe indeksowane Punkt takiego obrazu nie zawiera wprost informacji o kolorze, lecz o indeksie danego koloru w tablicy kolorów. Na etapie wyświetlania obrazu pobierany jest numer indeksu, na jego podstawie określane są wartości składowych koloru zapamiętane jako element tablicy kolorów o tym indeksie i dopiero wartości pobrane z tablicy stanowią o ostatecznym kolorze punktu. Obraz kolorowy z tablicą kolorów jest więc obrazem skalarnym, podobnie jak obraz w skali szarości. Dodatkowo istnieje odrębna struktura danych - tablica kolorów, która przechowuje informacje o wszystkich kolorach występujących na obrazie.

Pytanie 1 *Na czym polega różnica między typem danych obrazu a rodzajem obrazu?*

Do konwersji obrazu na inny typ danych wykorzystywane są następujące instrukcje:

Nazwa	Typ danych wyj.	Zakres zmienności	Dopuszczalne typy danych wej.
`im2uint8`	`uint8`	$< 0, 255 >$	`logical`,`uint16`,`double`
`im2uint16`	`uint16`	$< 0, 65535 >$	`logical` ,`uint8`,`double`
`mat2gray`	`double`	$< 0, 1 >$	`double`
`im2double`	`double`	$< -10^{308}, 10^{308} >$	`logical` ,`uint8`,`uint16`
`im2bw`	`logical`	$\{0, 1\}$	`uint8`,`uint16`,`double`

Pierwszy przykład do samodzielnego wykonania (Przykład 1) pozwala na zaznajomienie się z typami danych obrazów w MATLAB-ie. Ciąg znaków po symbolu `%` jest komentarzem i nie wymaga przepisywania. W tym przykładzie na końcach linii nie należy umieszczać średników - w ten sposób wynik operacji będzie widoczny na ekranie natychmiast po uruchomieniu odpowiedniej komendy.

Przykład 1 *Macierz obrazu*

```
m = [0 2 4 6 ; 6 8 10 12 ; 12 14 16 18 ; 18 20 22 24] % deklaracja macierzy
m = m*10 % zwiększenie wartości elementów macierzy
whos m  % wyświetlenie typu macierzy - ten typ nie może być obrazem
o1 = uint8(m) % konwersja na typ uint8
whos o1 % wyświetlenie typu macierzy - to już może być obraz
o2 = im2double(o1) % konwersja na typ double z zakresu <0,1>
whos o2 % wyświetlenie typu macierzy - porównaj z macierzą m!
```

Zadanie 1 *Poeksperymentuj w podobny do pokazanego w Przykładzie 1 z obrazem typu* `uint16`*.*

Szczególnym rodzajem obrazu jest obraz kolorowy. W zależności od jego typu (indeksowany/nieindeksowany) stosuje się inną reprezentację w przestrzeni roboczej MATLAB-a. Obraz nieindeksowany jest więc macierzą trójwymiarową $M \times N \times 3$ (gdzie M i N są rozmiarami obrazu)

składającą się z trzech macierzy dwuwymiarowych $M \times N$, z których każda odpowiada jednej składowej koloru. Obraz kolorowy tego typu może być reprezentowany przez typ danych *uint*8, *uint*16, lub *double*.

Obraz kolorowy indeksowany wymaga określenia dwóch macierzy: obrazu właściwego i tablicy kolorów. Sam obraz w tym przypadku może być jedynie typu `uint8` lub `uint16`. Tablica kolorów przechowująca składowe RGB jest na ogół tablicą o rozmiarach 3×256 (dla obrazu `uint8`) lub 3×65535 (dla obrazu `uint16`), której każdy element jest typu `double` z zakresu $< 0, 1 >$ i opisuje odpowiednią składową danego koloru.

Pytanie 2 *Czym różni się obraz kolorowy indeksowany od nieindeksowanego?*

Wszelkie ciągi instrukcji MATLAB-a można zapisywać w postaci funkcji. W celu zapisania funkcji należy otworzyć okno edytora, utworzyć pusty plik na następnie wpisać tekst funkcji. Pierwsza linia (nagłówek) funkcji zawsze powinna zawierać słowo kluczowe `function`, oraz zmienną wyjściową wraz z ciągiem argumentów wg. następującej składni:

Przykład 2 *Funkcja*

```
function zmienna_wyjsciowa = nazwa_funkcji(zmienna_wej1, zmienna_wej2, ...)

-- ciało funkcji --

end
```

Po wpisaniu tekstu funkcji należy ją zapisać, podczas zapisywania MATLAB zaproponuje nazwę funkcji taką jak została zdefiniowana w tekscie (`nazwa_funkcji`). Funkcja jest zapisywana w bieżącym katalogu roboczym[1]. Po zapisaniu funkcja może być wywoływana z linii komend w taki sam sposób jak funkcje predefiniowane MATLAB-a.

Przykładowa funkcja pozwalająca na dodanie dwóch macierzy jest następująca:

Przykład 3 *Przykładowa funkcja*

```
function c = dodajmac(a,b)
 c = a + b;
end
```

Po zapisaniu na dysku, w bieżącym katalogu pojawi się plik `dodajmac.m` zawierający tę funkcję. Funkcję tę, po zapisaniu można wywołać z linii komend np. w następujący sposób:

Przykład 4 *Wywołanie przykładowej funkcji*

```
q = [1 2 ; 3 4] % stwórz macierze
w = [1 1 ; 1 1]
dodajmac(q,w)   % wywołaj funkcję
```

2.2 Wczytywanie i zapisywanie obrazów

Obraz w środowisku MATLAB jest macierzą, która może być wpisywana ręcznie tak jak w przykładzie 1. Poza bardzo szczególnymi przypadkami takie obrazy nie mają większego zastosowania. Z oczywistych względów najpopularniejszą metodą tworzenia obrazów jest ich wczytywanie z plików graficznych bądź pobieranie bezpośrednio z kamery podłączonej do komputera. MATLAB obsługuje większość popularnych formatów graficznych, w tym: TIFF (`.tif,.tiff`), JPEG(`.jpg,.jpeg`), GIF(`.gif`), BMP(`.bmp`), PNG (`png`). Uniwersalną instrukcją wczytującą obraz z pliku graficznego

[1]Ścieżka dostępu do tego katalogu jest wyświetlona w górnej części głównego okna MATLAB-a. W oknie tym możliwa jest także ręczna zmiana bieżącego katalogu roboczego. W razie jakiegokolwiek niepowodzenia operacji na plikach należy w pierwszym rzędzie sprawdzić położenie katalogu roboczego. Bieżący katalog roboczy należy ustawić zgodnie ze wskazaniem prowadzącego ćwiczenie.

jest instrukcja `imread`. Jej argumentem jest nazwa pliku wraz ze ścieżką dostępu, zaś wartością zwracaną - macierz obrazu. Obrazu wczytywane są standartowo z bieżącego katalogu roboczego (*current directory*).

Typ danych macierzy obrazu tworzonej w trakcie działania instrukcji `imread` zależy od typu danych zapisanych w pliku graficznym. Typ ten można sprawdzić po wczytaniu obrazu instrukcją *whos*, której argumentem jest macierz obrazu. Można też sprawdzić go przed wczytaniem obrazu do pamięci komputera korzystając z instrukcji `imfinfo`, której argumentem jest nazwa pliku zawierającego obraz. Z uwagi na typowe parametry stosowane w zapisie obrazów w plikach graficznych, w większości przypadków wczytany obraz jest typu `uint8`. Ponieważ część instrukcji przetwarzania obrazów wymaga obrazów typu `double` jako argumentów, wczytany obraz typu `uint8` wymaga konwersji na typ `double`.

Instrukcją pozwalającą na zapisanie obrazu na dysku pod wskazaną nazwą jest `imwrite`, której pierwszym argumentem jest zmienna pod którą zapisany jest w pamięci obraz, zaś drugą - nazwa pliku. Istotne jest przy tym, by nazwa pliku zawierała rozszerzenie, gdyż na jego podstawie określany jest format zapisu pliku obrazu.

2.3 Wyświetlanie obrazów

Najprostszy sposób wyświetlenia korzysta z instrukcji `imshow`. Jej argumentem jest macierz obrazu, a efekcie na ekranie pojawia się nowe okno zawierające wyświetlany obraz.

Przykład 5 *Wczytywanie i wyświetlanie obrazów*

```
imfinfo('uscal.tif')  % wyświetlenie parametrów pliku graficznego
u = imread('uscal.tif');  % wczytanie obrazu - uwaga na średnik !
whos u % wyświetlenie parametrów macierzy obrazu
imshow(u) % wyświetlenie obrazu na ekranie
```

Zadanie 2 *W podobny sposób wczytaj, zbadaj i wyświetl obrazy szarościowe* `moon.tif`, `lotnicze.tif` *oraz kolorowe nieindeksowane* `baboon.jpg` *oraz* `kostka.bmp`. *Zwróć uwagę na informacje wyświetlane po wpisaniu komend* `imfinfo` *oraz* `whos`. *Jak na podstawie wyświetlonych danych stwierdzić, że mamy do czynienia z obrazami kolorowymi nieindeksowanymi?*

Komenda `imtool` pozwala na wyświetlanie obrazu z dodatkowymi możliwościami powiększania jego fragmentów, odczytu wartości pikseli wskazanych przez kursor oraz pomiarów odległości między wskazanymi pikselami obrazu. Składnia tej komendy jest identyczna ze składnią `imshow`.

Zadanie 3 *Zastosuj komendę* `imtool` *do wyświetlenia wybranych przez siebie obrazów.*

Sposób wczytywania i wyświetlania obrazów pokazany w Przykładzie 5 jest właściwy dla wszystkich rodzajów obrazów z wyjątkiem obrazów kolorowych indeksowanych. W przypadku tych ostatnich należy bowiem uwzględnić – oprócz samej macierzy obrazu – także i tablicę kolorów. Tablica ta jest podawana jako drugi argument komendy `imshow`. Nieuwzględnienie tablicy koloru spowoduje, że wyświetlony obraz będzie obrazem w skali szarości, o jasnościach odpowiadających indeksów w tablicy kolorów. Ponieważ obraz kolorowy indeksowany, z uwagi na ograniczoną liczbę kolorów, nie nadaje się do przetwarzania większością typowych metod, poddawany jest on zwykle konwersji na obraz nieindeksowany przy pomocy komendy `ind2rgb`, której argumentami są obraz oraz tablica kolorów. Pokazane jest to w przykładzie 6.

Przykład 6 *Wczytywanie i wyświetlanie obrazów kolorowych indeksowanych*

```
imfinfo('baboon1.gif')
[o tk] = imread ('baboon1.gif');
whos o  % wyświetlenie informacji o macierzy obrazu
whos tk % wyświetlenie informacji o tablicy kolorów
imshow (o) % czy tak powinien wyglądać nasz obraz?
```

```
imshow (o,tk) % a może tak?
o_rgb = ind2rgb(o,tk); % konwersja na typ nieindeksowany
whos o_rgb % porównaj z wynikami poprzednich komend 'whos'
imshow (o_rgb) % w tym przypadku tablica kolorów jest zbędna
```

Jak widać na Przykładzie 6 obraz indeksowany (w przykładzie – macierz `o`) wymaga przy wyświetlaniu podania tablicy kolorów (w przykładzie - macierz `tk`). W przeciwnym razie obraz zostanie wyświetlony jako w skali szarości o nierzeczywistych szarościach odpowiadających indeksom tablicy kolorów. Po konwersji na obraz nieindeksowany tablica kolorów nie jest już konieczna do wyświetlenia obrazu.

Zadanie 4 *W podobny sposób wczytaj, zbadaj, wyświetl i dokonaj konwersji obrazu indeksowanego* `mangan.bmp`.

Możliwe jest także wyświetlanie w pojedynczym oknie większej liczby obrazów. Najbardziej uniwersalny sposób takiego wyświetlania polega na zastosowaniu komendy `subplot(m,n,p)`. Parametry `m` oraz `n` określają liczbę obrazów odpowiednio w pionie i w poziomie, zaś `p` określa położenie aktualnego obrazu (licząc od pola lewego górnego do prawego dolnego). Metoda ta pozwala także na wyświetlanie innych rodzajów obiektów, np. wykresów, razem z obrazami. Alternatywnym sposobem wyświetlania wielu obrazów, ograniczonym jednak do obrazów tego samego typu oraz rozmiarów jest sposób polegający na utworzeniu z kilku macierzy obrazu – jednej, która następnie jest wyświetlana. Praktyczne zastosowanie obu rodzajów wyświetlania pokazuje przykład 7.

Przykład 7 *Wyświetlanie wielu obrazów*

```
im1 = imread('baboon.jpg');
im2 = imread('building.tif');
figure;
subplot(1,2,1);
imshow(im1);
subplot(1,2,2);
imshow(im2);
whos im1   % pokazuje typ obrazu - obraz nieindeksowany
im1r = im1(:,:,1);  % pierwsza składowa - czerwona
whos im1r  % typ obrazu - składowej czerwonej
im1g = im1(:,:,2);  % druga - zielona
im1b = im1(:,:,3);  % trzecia - niebieska
figure;  % nowy obraz
subplot(2,2,1);
imshow(im1r);
subplot(2,2,2);
imshow(im1g);
subplot(2,2,3);
imshow(im1b);
subplot(2,2,4);
imshow(im1);
figure;  % i jeszcze jeden obraz
imshow([im1r im1g im1b]);
figure;
imshow([im1r ; im1g ; im1b]);
figure;
imshow([im1r im1]);   % jaki jest efekt ?
```

Pytanie 3 *Dlaczego wywołanie* `imshow` *w ostatniej linii przykładu 7 spowodowało błąd ?*

2.4 Histogram

Istotną informację o obrazie zawiera jego histogram. Dla obrazów cyfrowych jest on funkcją, której argument stanowi odcień szarości punktu, zaś wartość - liczba punktów obrazu charakteryzujących się tą właśnie szarością. Histogram pozwala na określenie dynamiki obrazu i często jest wykorzystywany do określania parametrów wykorzystywanych w operacjach przetwarzania obrazów. Do wyznaczania histogramu obrazu w skali szarości wykorzystuje się instrukcję `imhist`, której argumentem jest obraz. Dla obrazu kolorowego wyznacza się histogramy jasności każdej ze składowych osobno. Rezultatem działania instrukcji `imhist` jest wektor wyjściowy, którego wartości odpowiadają liczbie punktów obrazu o jasności równej indeksom wektora zwiększonym o 1. Tak więc dla obrazu `uint8` lub `uint16` element wektora histogramu o indeksie 1 zawiera liczbę punktów obrazu o wartości 0, o indeksie 2 - liczbę punktów obrazu o wartości 1 itd. Długość wektora histogramu zależy od liczby odcieni szarości obrazu i wynosi dla obrazu `uint8` - 256, zaś dla `uint16` - 65536. Wywołanie instrukcji `imhist` bez podawania wektora wyjściowego powoduje wyświetlenie w odrębnym oknie histogramu w postaci graficznej.

Przykład 8 *Wyznaczanie histogramu obrazu*

```
o = imread('lotnicze.tif');
h = imhist(o);  % zapisz histogram jako wektor
whos h
subplot(1,2,1);
imhist(o) % wyświetl histogram
subplot(1,2,2);
imshow(o);
```

2.5 Profile obrazów

Profil obrazu w skali szarości jest dwuwymiarową funkcją pokazującą odcienie szarości punktów leżących wzdłuż zadanej linii prostej. Profil jest wektorem zawierającym kolejne punkty leżące wzdłuż zadanej linii.

Przykład 9 *Profile obrazów*

```
im = imread('kalkulator.tif');
imshow(im)
px=[0 1360];  % wsp. x początku i końca linii
py=[385 385]; % wsp. y początku i końca linii
line(px,py,'Color',[1 1 1]);  % rysowanie linii przekroju na obrazie oryginalnym
pr = improfile(im,px,py);
figure;    % nowe okno wyświetlania
plot(pr)   % rysuj wykres
```

Zadanie 5 *Wyznacz profile wzdłuż linii prostych o innym początku i końcu. Sprawdź czy zmienność funkcji profilu odpowiada zmienności odcieni szarości na obrazie.*

Zadanie 6 *Napisz funkcję, która dla trzech danych wejściowych: obrazu, wektorów px i py (rozumianych tak jak w przykładzie 9) wyświetla w pojedynczym oknie obraz początkowy z linią przekroju (z lewej strony) oraz wykres profilu (z prawej).*

W przypadku obrazów kolorowych, macierzą wyjściową komendy `improfile` jest macierz o trzech kolumnach. W każdej z nich jest zawarty profil wyznaczony na kolejnych składowych.

Komenda `improfile` pozwala również na interaktywne wskazywanie linii profilu obrazu. Należy ją w tym celu wywołać bez argumentów. Wówczas w ostatnio wyświetlonym oknie obrazu będzie

można przy pomocy kursora myszy wyznaczyć linię profilu. Linia taka może być linią łamaną. W celu rozpoczęcia rysowania linii należy kliknąć lewym klawiszem myszki w punkcie początku linii. Następnie kolejnymi klinięciami wskazuje się kolejne punkty krzywej. Dwukrotne kliknięcie kończy akcje zaznaczania linii profilu.

Interaktywne wyznaczanie profilu obrazu kolorowego pokazuje przykład 10.

Przykład 10 *Interaktywne wyznaczanie profilu obrazu kolorowego*

```
im = imread('baboon.jpg');
imshow(im);
p = improfile;  % interaktywne wczytywanie linii przekroju
plot(p(:,:,1),'r');
hold;
plot(p(:,:,2),'g');
plot(p(:,:,3),'b');
```

Zadanie 7 *Na bazie funkcji utworzonej w zadaniu 6, utwórz nową funkcję wykonującą to samo zadanie ale dla obrazów kolorowych.*

3 Punktowe metody poprawy jakości obrazów

Każda operacja przetwarzania obrazów wymaga wyznaczenia kolejno wartości każdego punktu na *obrazie wyjściowym* na podstawie wartości punktów *obrazu wejściowego*. W punktowych metodach przetwarzania obrazu wartość piksela o danych współrzędnych na obrazie wyjściowym zależy tylko i wyłącznie od wartości piksela o tych samych współrzędnych na obrazie wejściowym. W tego typu przekształceniach nie jest brane pod uwagę sąsiedztwo punktu. Do wykonywania operacji punktowych stosuje się często tablicę korekcji (ang. look-up-table, LUT). Tablica ta jest w istocie wektorem i zawiera wartości punktów na obrazie wyjściowym dla wszystkich dopuszczalnych wartości punktów na obrazie wejściowym. W przypadku obrazu w skali szarości 8-bitowego, gdzie wyróżnia się 256 odcieni szarości, tablica taka jest wektorem o tylu właśnie elementach. Wartość elementu tego wektora o indeksie i wskazuje na wartość jaka zostanie przypisana na obrazie wyjściowym każdemu pikselowi, którego wartość na obrazie wejściowym wynosi i.

Operacje punktowe mogą być opisane przy pomocy *krzywych tonalnych*. Krzywe te opisują zależność wartości punktów obrazu wejściowego i wyjściowego. W praktycznej realizacji krzywe te są kodowane w postaci *tablic korekcji*, gdzie dla każdej możliwej wartości piksela obrazu wejściowego zapisana jest wartość, jaką otrzymuje piksel na obrazie wyjściowym.

Dla obrazów kolorowych operacje tego typu wykonuje się zwykle na każdej składowej koloru z osobna.

3.1 Negatyw obrazu

Najprostszą operacją typu punktowego jest negatyw obrazu, polegający na odjęciu wartości każdego piksela od dopuszczalnej, maksymalnej wartości punktu obrazu. Najwygodniejszym sposobem wyznaczenia negatywu obrazu jest zastosowanie komendy `imcomplement`, której argumentem jest obraz, a wartością zwracaną macierz obrazu negatywowego.

Zadanie 8 *Wczytaj, wyznacz negatywy i wyświetl je dla następujących obrazów wejściowych* `baboon.jpg`, `kostka.bmp`, `uscal.tif`, `lotnicze.tif`

Jedną z alternatywnych możliwości wykonania negatywu obrazu jest bezpośrednie zastosowanie tablicy korekcji.

Przykład 11 *Zastosowanie tablicy korekcji – negacja*

```
im  = im2uint8(imread('lotnicze.tif'));
lut1 = 0:255; % macierz lut nie zmieniająca wartości pikseli
im2 = uint8(lut1(double(im)+1));
subplot(2,2,[1 3]); imshow(im);
subplot(2,2,2); imshow(im2);
subplot(2,2,4); plot(lut1);
lut2 = 255 - lut1;
im3 = uint8(lut2(double(im)+1));
figure;
subplot(2,2,[1 3]); imshow(im3);
subplot(2,2,2); imshow(im);
subplot(2,2,4); plot(lut2);
```

Wykres, pokazany w oknie wynikowym Przykładu 11, przedstawia zależność wartości pikseli obrazu wyjściowego od wartości pikseli obrazu wejściowego. Wykres taki nosi nazwę krzywej tonalnej.

3.2 Operacje arytmetyczne na pojedynczym obrazie

Do najprostszych operacji punktowych zaliczamy także operacje artymetyczne, polegające na (w nawiasach – odpowiednie komendy):

- dodaniu (instrukcja `imadd`),
- odjęciu (`imsubtract`),
- pomnożeniu (`immultiply`) lub
- podzieleniu (`imdivide`)

wartości każdego punktu przez ustaloną wielkość, będącą drugim argumentem danej operacji (pierwszym jest obraz). Instrukcje wykonujące operacje arytmetyczne są dwuargumentowe. Pierwszym argumentem jest obraz wejściowy, drugim zaś wartość liczbowa. W sytuacji, gdy po wykonaniu operacji dla konkretnego piksela otrzymana wartość znajdzie się poza dopuszczalnym zakresem, zostanie ona obcięta do najbliższej wartości z zakresu. Przykładowo, dla operacji dodawania do obrazu liczby 50 i piksela obrazu `uint8` o wartości 240, zamiast wartości 290 punktowi obrazu wyjściowego przypisana zostanie wartość 255.

Przykład 12 *Proste operacje arytmetyczne – dodawanie*

```
o = imread('lotnicze.tif');
o1 = imadd (o, 50);
subplot(1,2,1); subimage(o); % wyświetlanie kilku obrazów w pojedynczym oknie
subplot(1,2,2); subimage(o1);
```

Przykład 13 *Operacje arytmetyczne – zastosowanie tablicy korekcji*

```
im  = im2uint8(imread('lotnicze.tif'));
lut1 = 0:255; % macierz lut nie zmieniająca wartości pikseli
lut2 = lut1 + 70;
lut2 = lut2 .* (lut2 < 256) + 255.*(lut2 > 255)
im2 = uint8(lut2(double(im)+1));
figure;
subplot(2,2,[1 3]); imshow(im2);
subplot(2,2,2); imshow(im);
subplot(2,2,4); plot(lut2);
```

```
lut3 = lut1 - 50;
lut3 = lut3 .* (lut3 >= 0);
im3 = uint8(lut3(double(im)+1));
figure;
subplot(2,2,[1 3]); imshow(im3);
subplot(2,2,2); imshow(im);
subplot(2,2,4); plot(lut3);
```

Zadanie 9 *Wykonaj w podobny do pokazanego w Przykładzie 13 sposób operację mnożenia i dzielenia obrazu przez stałą wartość. Obejrzyj odpowiednie krzywe tonalne.*

Zadanie 10 *Wykonaj w dowolny sposób operacje odejmowania, mnożenia i dzielenia obrazów dla obrazu* `lotnicze.tif` *oraz* `baboon.jpg`*. Wyświetl je w pojedynczym oknie zawierającym 4 obrazy.*

Pytanie 4 *Jakie efekty dały poszczególne operacje arytmetyczne ? Które z nich spowodowały rozjaśnienie obrazu, a które przyciemnienie ? Jakie efekty uboczne zaistniały przy okazji wykonywania tych operacji ?*

Pytanie 5 *Jak dotychczas omówione operacje (negacja, dodawanie, odejmowanie, mnożenie i dzielenie) wpływają na histogram obrazu? Wykonaj stosowne eksperymenty, obejrzyj histogramy przed i po wykonaniu operacji.*

3.3 Korekcja gamma i rozciąganie histogramu

Bardzo przydatną kolejną operacją jest korekcja obrazu z wykorzystaniem instrukcji `imadjust`. Łączy ona w sobie rozciąganie zakresu zmienności punktów obrazu (rozciąganie histogramu) z korekcją gamma. Pierwsze z wymienionych działań pozwala na zwiększenie dynamiki obrazu. Często zdarza się, że obraz sprawia wrażenie bladego przez to, że brakuje w nim skrajnych odcieni szarości - zbliżonych do bieli i/lub czerni. W takim przypadku stosuje się przekształcenie rozszerzające skalę odcieni szarości występujących w obrazie w ten sposób, by rozpoczynała się od pełnej czerni (wartość punktu - 0) aż do pełnej bieli (wartość punktu w obrazie `uint8` - 255) - rozciąganie histogramu. Takie przekształcenie zwiększa kontrast obrazu przez zwiększenie odstępów między kolejnymi jasnościami występującymi w obrazie.

Korekcja gamma, z kolei, pozwala na poprawę kontrastu obszarów jasnych kosztem ciemnych lub odwrotnie. Operację tę stosuję się w sytuacji, gdy jasne obszary obrazu są zbyt mało kontrastowe, zaś ciemne nie odgrywają większej roli, lub odwrotnie. Składnia instrukcji `imadjust` jest następująca:

```
obraz_wyj = imadjust(obraz_wej,[dolna_wej gorna_wej],[dolna_wyj gorna_wyj], gamma);
```

przy czym `[dolna_wej gorna_wej]` jest zakresem jasności na obrazie wejściowym, który jest przekształcany w zakres `[dolna_wyj gorna_wyj]` na obrazie wyjściowym. Niezależnie od typu obrazu wejściowego wszystkie cztery wielkości powinny być z zakresu $< 0, 1 >$, gdzie 0 odpowiada najniższemu poziomowi jasności, zaś 1 - najwyższemu. Parametr `gamma` jest opcjonalny i określa współczynnik wykorzystywany w korekcji gamma. Dla `gamma<1` poprawiany jest kontrast ciemniejszych fragmentów obrazu kosztem jaśniejszych, zaś dla `gamma>1` - odwrotnie. Dla `gamma=1` korekcja gamma nie zmienia rezultatu i wtedy parametr ten może być pominięty.

Przykład 14 *Korekcja obrazu instrukcją* `imadjust`

```
o = imread('lotnicze.tif');
subplot(2,2,1); subimage(o);
```

```
subplot(2,2,2); subimage(imadjust(o,[0.2 1],[0 1]));
subplot(2,2,3); subimage(imadjust(o,[0.4 1],[0 1]));
subplot(2,2,4); subimage(imadjust(o,[0.2 1],[0.1 0.6]));
figure;   % utwórz nowe okno wyświetlania obrazów
subplot(2,2,1); subimage(o);
subplot(2,2,2); subimage(imadjust(o,[0 1],[0 1],0.5));
subplot(2,2,3); subimage(imadjust(o,[0 1],[0 1],2));
subplot(2,2,4); subimage(imadjust(o,[0 1],[0 1],5));
```

Zadanie 11 *Wykonaj korekcję obrazu z przykładu 14 dla różnych parametrów wejściowych funkcji* `imadjust`*. Jak parametry wpływają na wynik operacji? Sprawdź jak zmienia się histogram obrazu wynikowego dla różnych parametrów ?*

Zadanie 12 *Dotychczas omówione operacje punktowe mogą być także wykonane dla obrazów kolorowych nieindeksowanych. Sprawdź, jakie efekty daje wykonanie wszystkich omówionych operacji na obrazie* `kostka.bmp`*.*

Pytanie 6 *W efekcie których omówionych do tej pory operacji następuje rozjaśnianie obrazu, czym różnią się poszczególne rodzaje rozjaśnienia ?*

Pytanie 7 *W efekcie których operacji następuje przyciemnianie obrazu, czym różnią się poszczególne rodzaje przyciemniania ?*

Pytanie 8 *Jak wyglądają krzywe tonalne dla operacji arytmetycznych z przykładu 12 i zadania 10 ?*

Pytanie 9 *Jak korekcja gamma wpływa na histogram obrazu? Wykonaj stosowne eksperymenty, obejrzyj histogramy przed i po wykonaniu operacji.*

3.4 Wyrównywanie histogramu

Inną operacją wykorzystującą histogram obrazu jest operacja wyrównywania histogramu. Jej celem jest uzyskanie histogramu jak najbliższemu tzw. histogramowi wyrównanemu. Idealnym histogramem wyrównanym jest taki, którego wykres jest linią prostą równoległą do osi „x". Obraz o histogramie wyrównanym charakteryzuje się tym, że występują w nim punkty o wszystkich możliwych wartościach oraz że liczba punktów każdej z tych wartości jest taka sama. Przykładowo, dla 8-bitowego obrazu w skali szarości o rozmiarach 100x100 pikseli o idealnie wyrównanym histogramie, wszystkie punky histogramu mają tę samą wartość wynoszącą $\frac{10000}{256}$. W rzeczywistości taka sytuacja występuje niezwykle rzadko. Nie jest także możliwe uzyskanie histogramu idealnie wyrównanego przy pomocy operacji punktowej. Łatwo jest bowiem sobie wyobrazić sytuację, gdy liczba punktów obrazu o pewnym odcieniu szarości jest większa niż wartość w histogramie wyrównanym. Dlatego w metodzie punktowej wyrównywania histogramu dąży się do tego, by histogram obrazu wynikowego był jak najbliższy histogramowi wyrównanemu, ale niekoniecznie mu równy. Operację wyrównywania histogramu realizujemy korzystając z funkcji `histeq`.

Przykład 15 *Korekcja obrazu instrukcją* `histeq`

```
I = imread('tire.tif');
J = histeq(I);
subplot(2,2,1);
imshow(I);
```

```
subplot(2,2,2);
imshow(J);
subplot(2,2,3);
imhist(I,64);
subplot(2,2,4);
imhist(J,64);
```

Zadanie 13 *Czy podobny do uzyskanego w Przykładzie 15 efekt końcowy można otrzymać stosując* `imadjuest` *? Zastanów się dlaczego ?*

Zadanie 14 *Wykonaj przykład "Contrast Enhancement Techniques" dostępny w systemie pomocy do Image Processing Toolbox.*

4 Operacje punktowe z większą liczbą obrazów

Operacje na dwóch obrazach wymagają zwykle obrazów wejściowych o takich samych rozmiarach i tego samego typu. W tego typu operacjach wartość piksela obrazu wyjściowego o danych współrzędnych jest wyznaczana na podstawie punktów o tych samych współrzędnych na dwóch lub więcej obrazach wejściowych. Najczęściej stosowanymi operacjami tego typu są operacje porównywania obrazów, operacje arytmetyczne, operacje punktowego minimum i maksimum. czy też operacje pozwalające na ukrywanie jednego obrazu w drugim. Inną kategorią tego typu operacji są punktowe przekształcenia pojedynczego obrazu w większą ich liczbę, np. generowanie trzech składowych obrazu kolorowego na podstawie obrazu w skali szarości jak również metody przekształcające oryginalne składowe kolorów w inne, co ma zastosowanie w konwersji przestrzeni barw obrazu.

4.1 Operacje arytmetyczne

Do operacji arytmetycznych na dwóch obrazach zalicza się:

- dodawanie (komenda `imadd`),
- odejmowanie (`imsubtract`),
- mnożenie (`immultiply`) lub
- dzielenie (`imdivide`)

dwóch obrazów będących argumentami odpowiednich funkcji.

Przydatną instrukcją jest `imabsdiff` wyznaczająca bezwzględną różnicę punktową dwóch obrazów. Wynikiem tej operacji jest obraz o takich samych rozmiarach jak obrazy wejściowe. Każdy punkt tego obrazu zawiera bezwzględną wartość różnicy między odpowiednimi punktami obrazów wejściowych.

Zadanie 15 *Znajdź (lub odpowiednio przekształć) dwa obrazy kolorowe tych samych rozmiarów i typów. Wykonaj na nich wszystkie wymienione operacje.*

4.2 Punktowe minimum i maksimum

Operacje punktowego minimum i makimum, dla każdego punktu obrazu wyjściowego wyznaczają wartość odpowiednio mniejszą lub większą spośród wartości punktów obrazów wejściowych o tych samych współrzędnych. *Obraz minimalny* powstaje przez przypisanie każdemu jego punktowi wartości minimalnej spośród wartości punktów o tych samych współrzędnych obrazów wejściowych.

Dla obrazów binarnych (X, Y) odpowiednikiem obrazu minimalnego jest iloczyn logiczny $X \cap Y$ zbiorów punktów należących do obu obrazów. *Obrazem maksymalny* nazywamy obraz powstający w identyczny sposób, ale z wyznaczeniem wartości maksymalnej spośród wartości odpowiednich pikseli obrazów wejściowych. Dla obrazów binarnych (X, Y) odpowiednikiem obrazu maksymalnego jest suma logiczna $X \cup Y$ zbiorów punktów należących do obu obrazów.

Operacje te mogą służyć m.in. do nakładania części jednego obrazu na inny.

Przykład 16 *Nakładanie obrazów zgodnie z maską*

```
im1 = imread('baboon.jpg');
[im2 p2] = imread('lena.tif');
im2a = im2uint8(ind2rgb(im2,p2)); % konwersja typu obrazu
imshow([im1 im2a]); title('Obrazy początkowe');
im3 = zeros(512,512);
im3(200:399,200:399)=ones(200,200);
im3a = im2uint8(im3);
im3b(:,:,1) = im3a;   % maska w wersji kolorowej - składowa R
im3b(:,:,2) = im3a;   % składowa G
im3b(:,:,3) = im3a;   % składowa B
im3c = imcomplement(im3b); % odwrócona maska
figure;
imshow([im3b im3c]); title('Obraz - maska');
im5 = min(im2a, im3b);
im4 = min(im1, im3c);
im6 = max(im4,im5);
figure;
imshow([im4 im5 im6]); title('Nałożenie');
```

Zadanie 16 *Napisz funkcję,* `nakladaj(imin1,imin2,immask)`*, która dla trzech argumentów wejściowych: dwóch obrazów kolorowych nieindeksowanych (*`imin1,imin2`*) oraz obrazu binarnego - maski (*`immask`*), zwraca obraz, w którym na pierwszy obraz kolorowy wejściowy jest nałożony drugi. Nałożenie powinno wyglądać w ten sposób, że punkty dla których maska jest równa 0 są pobierane z* `imin1`*, zaś punkty o masce równej 1 - z* `imin2`*.*

Przykład 17 jest podobny do przykładdu 16. Tym razem jednak obszar wycinany jest określany interaktywnie przez użytkownika przy pomocy komendy `roipoly`. Ponadto, przykład wykorzystuje funkcję `nakladaj`, którą należało napisać w Zadaniu 16.

Przykład 17 *Nakładanie obrazów zgodnie z maską definiowaną interaktywnie*

```
im1 = imread('baboon.jpg');
[im2 p2] = imread('lena.tif');
im2a = im2uint8(ind2rgb(im2,p2)); % konwersja typu obrazu
imshow(im1*0.5 + im2a*0.5); title('Obrazy początkowe');
im3 = roipoly(im1*0.5 + im2a*0.5);
im4 = nakladaj(im1,im2a,im3);
figure;
imshow([im1 im2a im4]); title('Nałożenie');
```

4.3 Mieszanie liniowe

Inną operacją pozwalającą na połączenie dwóch obrazów jest *mieszanie liniowe*. Jego działanie polega na obliczeniu kombinacji liniowej kolejno dla wszystkich pikseli dwóch obrazów i uzyskaniu w ten sposób obrazu wynikowego. Kombinacja wyznaczana jest ze współczynnikami wagowymi stałymi dla wszystkich pikseli każdego obrazu. Można ją zapisać przy pomocy następującego równania:

$$g(p) = f_1(p) \cdot (1 - \alpha) + f_2(p) \cdot \alpha \tag{1}$$

przy czym f_1,f_2 są obrazami wejściowymi (mieszanymi); g - jest wynikiem mieszania liniowego, a α jest współczynnikiem wagowym ($0 \leq \alpha \leq 1$).

Efektem mieszania liniowego jest obraz zawierający wzajemnie przenikające się obrazy wejściowe. O widoczności poszczególnych obrazów na obrazie wynikowym decyduje parametr α. Warunek normalizacyjny nałożony na ten współczynnik powoduje iż zbiór wartości pikseli obrazu wynikowego pokryje się ze zbiorem wartości pikseli obrazów wejściowych.

Szczególnym przypadkiem mieszania liniowego jest obraz uśredniony, występujący gdy $\alpha = 0,5$.

Wyznaczając obrazy interpolowane zgodnie z równaniem (1) dla rosnących wartości α otrzyma się w efekcie sekwencję interpolującą. Sekwencja ta będzie zawierała przekształcenie jednego obrazu wejściowego (f_1) w drugi (f_2).

Przykład 18 pokazuje przykładowe mieszanie liniowe dwóch obrazów kolorowych.

Przykład 18 *Mieszanie liniowe*

```
im1 = imread('baboon.jpg');
im1a = im2double(im1);
[im2 p2] = imread('lena.tif');
im2a = ind2rgb(im2,p2);
imshow([im1a im2a]); title('Obrazy początkowe');
m2 = 0.2*im1a + 0.8*im2a;
m4 = 0.4*im1a + 0.6*im2a;
m6 = 0.6*im1a + 0.4*im2a;
m8 = 0.8*im1a + 0.2*im2a;
figure;
imshow([im2a m2 m4; m6 m8 im1a]); title('Sekwencja');
```

Zadanie 17 *Mieszanie liniowe może być wykonane także przy pomocy wbudowanie komendy MALAB-a* `imlincomb`*. Sprawdź z systemie pomocy MATLAB-a w jaki sposób należy wywołać tę komendę i wykonaj z jej zastosowaniem Przykład 18.*

Kolejne klatki sekwencji mogą być także zapamiętywane w pojedynczej macierzy czterowymiarowej (każda klatka sekwencji to macierz trójwymiarowa zawaierająca obraz kolorowy). Klatki tak zapisanej sekwencji można wyświetlić komendą `montage`. Taki zapis sekwencji umożliwia także jej wyświetlene w jeszcze inny sposób - w postaci filmu (klatka po klatce). W tym celu należy zastosować komendy `immovie` oraz `movie`. Pierwsza z nich tworzy specjalną strukturę danych, która może być następnie wyświetlona przy pomocy drugiej z wymienionych komend. Komenda `movie(struktura_obr,liczba_powtorzen, liczba_klatek_na_sek)` wyświetla sekwencję dla zadanej liczby powtórzeń oraz liczbie wyświetlanych klatek na sekundę. Standartowo, komenda `movie` nie dostosowuje rozmiaru okna, w którym jest wyświetlana sekwencja do rozmiaru klatki tej sekwencji. Należy więc zrobić to niezależnie, w sposób pokazany w przykładzie 19.

Przykład 19 *Wyświetlanie sekwencji obrazów*

```
s(:,:,:,1)=im2a;
s(:,:,:,2)=m2;
s(:,:,:,3)=m4;
s(:,:,:,4)=m6;
s(:,:,:,5)=m8;
s(:,:,:,6)=im1a;
montage(s) ; % klatki sekwencji
sx = size(im1a,1)  % rozmiar x klatki
sy = size(im1a,2)  % rozmiar y klatki
figure;
```

```
m = immovie(s);
figure('position',[100 100 (100+sx) (100+sy)]);
movie(m,4,1);
```

Zadanie 18 *Napisz funkcję* `sekwencja(im1,im2,n)`*, która dla dwóch obrazów początkowych* `im1` *oraz* `im2` *zwraca czterowymiarową macierz zawierającą sekwencję mieszania liniowego składającą się z* `n` *klatek. W funkcji wykorzystaj pętlę* `for`*. Informacje o sposobie użycia tej konstrukcji języka MATLAB znajdź w systemie pomocy.*

4.4 Prosta steganografia

Steganografia jest nauką o ukrywaniu informacji. Obrazy cyfrowe stanowią bardzo wygodny nośnik informacji ukrytej. Najprostszy sposób ukrycia informacji w obrazie wykorzystuje ograniczenia oka ludzkiego do rozróżniania odcieni szarości i barw o zbliżonych parametrach. Dla obrazów w skali szarościch zdolność oka ludzkiego do rozróżniania sąsiadujących odcieni szarości kończy się na skali od czystej bieli do czystej czerni składającej się z około 80 odcieni szarości. Oznacza to, że przy większej liczbie pośrednich odcieni szarości nie jest możliwe rozróżnienie odcieni sąsiadujących na skali. W typowej, 8-bitowej skali liczba różnych odcieni wynosi 256. Biorąc pod uwagę wcześniejsze uwagi oznacza to, że w przypadku takiej skali szarości nie jest możliwe rozróżnienie dowolnych dwóch, trzech, a nawet czterech kolejnych odcieni szarości. Dlatego, spośród 8 bitów składających się na każdy piksel, w istocie odbierane przez zmysł wzroku człowieka jest 6 najbardziej znaczących. Dwa najmniej znaczące moga mieć dowolna wartość i nie zostanie to zauważone przez obserwatora.

Operacje bitowe są operacjami logicznymi, gdzie każda liczba jest traktowana jako ciąg bitów. Są to klasyczne operatory logicznej koniunkcji („i") `bitand` i alternatywy („lub") `bitor` oraz negacji `bitcmp`. Inną przydatną operacją bitową jest operacja przesuwania bitów w lewo `bitshift`.

Przykład 20 *Operacje bitowe na liczbach*

```
v = 115
dec2bin(v)  % konwersja liczby dziesiętnej na binarną
w = 222
dec2bin(w)
a = bitand(v,w)
dec2bin(a)
b = bitor(v,w)
dec2bin(b)
c = bitcmp(v)
dec2bin(c)
d = bitshift(v,2);
dec2bin(d)
```

W przypadku obrazów operacji bitowej na obrazie w skali szarościm lub kolorowym operacje te wykonuje się na poszczególnych pikselach obrazu. Jest to jednak możliwe jedynie dla całkowitoliczbowej reprezentacji obrazu tj. dla obrazów typu `uint8` lub `uint16`.

Przykład 21 *Operacje bitowe na pikselach obrazu*

```
im = imread('lenagray.bmp');
im1 = bitand(im,bin2dec('10000000')); % koniunkcja bitowa
im2 = bitcmp(im); % negacja
im3 = bitor(im,bin2dec('00100000')); % alternatywa bitowa
imshow([im im1 ; im2 im3]);
```

Przykład 22 *Bity a percepcja obrazu*

```
im = imread('lenagray.bmp');
im7 = bitand(im,bin2dec('11111110'));
im6 = bitand(im,bin2dec('11111100'));
im5 = bitand(im,bin2dec('11111000'));
im4 = bitand(im,bin2dec('11110000'));
im3 = bitand(im,bin2dec('11100000'));
imshow([im im7 im6 ; im5 im4 im3]);
```

Pytanie 10 *Jakie operacje zostały wykonane w przykładzie 22 ? Co pokazują ich wyniki ?*

Z uwagi na brak postrzegania różnic na dwóch najmniej znaczących bitach możliwe dowolne ich ustawienie. Można więc zakodować na nich dowolną informację zarówno tekstową jak i obrazową.

Przykład 23 *Ukrywanie obrazu w obrazie*

```
im = imread('lenagray.bmp');
b = imread('leters1.bmp');
whos im
whos b
im2 = imresize(im,0.5);
whos im2 % obrazy mają takie same rozmiary
imshow([im2 b]);
b1 = bitand(b,bin2dec('00000001'));  % interesuje nas najmniej znaczący bit (LSB)
im3 = bitand(im2,bin2dec('11111110')); % zerujemy LSB w im2
im4 = bitor(b1,im3);
imshow([im3 im4]); % czy widac roznice ?
imshow([im4 bitand(im4,1)*128]) % a jednak ...
```

Zadanie 19 *Powtórz eksperyment z Przykładu 23 kodując obraz na innym bicie, np. czwartym. Czy i tym razem obraz ukryty jest niewidoczny ? Dlaczego ? Wskazówka: zmiany wymagają jedynie linie zaczynające się od* `b1 = ...`, `im3 = ...` *oraz ostatnia linia* `imshow`.

4.5 Konwersja przestrzeni barw

Najczęściej spotykaną przestrzenią barwną jest przestrzeń RGB, gdzie trzy składowe koloru odpowiadają barwom czerwonej, niebieskiej i zielonej. Przestrzeń ta jest wykorzystywana w różnego rodzaju urządzeniach wyświetlających obraz jak również w niektórych standardach zapisu obrazów. Istnieje także jednak wiele innych przestrzeni barw o odmiennej interpretacji poszczególnych kanałów. Przykład 24 ilustruje konwersję z przestrzeni RGB do przestrzeni HSV.

Przykład 24 *Konwersja przestrzeni barw*

```
im = imread('baboon.jpg');
imshow([im(:,:,1), im(:,:,2), im(:,:,3)])
im2 = rgb2hsv(im);
imshow([im2(:,:,1), im2(:,:,2), im2(:,:,3)])
```

Pytanie 11 *Jakie cechy barw są reprezentowane przez poszczególne składowe w przestrzeni HSV ?*

4.6 Pseudokolorowanie

Pseudokolorowanie polega na konwersji obrazu w skali szarości na obraz kolorowy w celu poprawy jego czytelności. Jego celem jest nie tylko uatrakcyjnienie wyświetlanego obrazu, ale także zwiększenie jego czytelności. To ostatnie jest związane z faktem ograniczonej zdolności oka ludzkiego

do rozróżniania niewiele różniących się jasności. Pseudokolorowanie polega na zastosowaniu trzech tablic korekcji na pojedynczym obrazie w skali szarości w celu otrzymania trzech różniących się od siebie składowych barwnych. Metoda jest naturalnym sposobem wyświetlania obrazów z kolorem indeksowanym. Tablica (paleta) kolorów jest osobną strukturą danych, zawierającą w istocie trzy tablice korekcji. Stosując różne palety kolorów możliwe jest wyświetlanie obrazu w różnych skalach barwnych. Image Processing Toolbox oferuje kilka predefiniowanych tablic kolorów, które możemy uzyskać stosując komendę `colormap`. Możliwe jest też samodzielne zdefiniowanie tablicy kolorów przez użytkownika. Wygodnym sposobem wyświetlenia krzywych tonalnych przypisanych do poszczególnych składowych koloru jest funkcja `rgbplot`.

Przykład 25 *Pseudokolorowanie*

```
im = imread('gray.bmp');
im2 = imread('lenagray.bmp');
imshow([im im2]) % wyświetlenie obrazów w trybie w skali szarościm
col = colormap('jet') % jaką macierzą jest tablica kolorów (rozmiar, typ danych) ?
subplot(1,2,1)
rgbplot(col);
subplot(1,2,2)
imshow([im; im2],col); % wyświetlenie obrazów w trybie pseukolorowania
```

Zadanie 20 *Sprawdź jakie inne tablice kolorów są dostępne w Matlab-ie. Wykonaj pseudokolorowanie obrazu 'CHAOS2.BMP' z różnymi paletami kolorów.*

5 Punktowe metody segmentacji obrazów

Celem segmentacji obrazu jest takie jego przekształcenie, by na obrazie wynikowym obszary obrazu reprezentujące odrębne elementy sceny wizyjnej były jednoznacznie opisane albo poprzez przypisanie im jednolitej wartości piksela (segmentacja obszarowa) albo poprzez otoczenie ich jednoznaczną linią konturową (segmentacja konturowa). W przypadku obrazów zawierających elementy sceny wizyjnej wyróżniające się spośród innych pikseli obrazu charakterystycznym zakresem zmienności odcieni szarości lub barw, segmentacja jest możliwa poprzez określenie warunków (zakresu zmienności wartości) jakie powinna spełniać wartość piksela przynależącego do danego obiektu. Tego typu segmentacja jest więc operacją typu punktowego.

5.1 Progowanie jasności obrazu

Progowanie jasności polega na przetworzeniu oryginalnego w taki sposób, by punkty, których wartość mieści się w zadanym zakresie, otrzymały wartość 1, pozostałe zaś – wartość 0. Dobór progu (lub progów) segmentacji może być ustawiony przez projektanta (empirycznie) lub automatycznie. Przy danym progu operacja sprogowania obrazu może być wykonana na kilka sposobów. Pierwszy z nich korzysta z instrukcji `im2bw`, której pierwszym argumentem jest progowany obraz, zaś drugim – próg. Niezależnie od typu obrazu, próg zawsze jest wartością rzeczywistą z zakresu $< 0, 1 >$. Inny sposób progowania polega na zastosowaniu bezpośrednio operatora nierówności $<$ lub $>$. Wówczas wartość progowa powinna być tego samego typu, co punkty obrazu. Oba warianty pokazuje Przykład 26.

Przykład 26 *Segmentacja przez progowanie*

```
>> o = imread('dowels.tif');
>> subplot(1,3,1); subimage(o);
>> seg = im2bw(o,0.5);    % pierwszy sposób progowania
>> subplot(1,3,2); subimage(seg);
>> seg2 = o > 128;        % drugi sposób progowania
```

```
>> subplot(1,3,3); subimage(seg2);
```

Zadanie 21 *Dobierz eksperymentalnie najlepszą wartość progową tj. taką, przy której wynik segmentacji jest najlepszy.*

Przykład 27 *Segmentacja przez progowanie - automatyczny dobór progu*

```
>> o = imread('dowels.tif');
>> t = graythresh(o)
>> seg = im2bw(o,t);
>> imshow(seg);
```

Progowanie z automatycznym doborem wartości progowej zostało zaimplementowane w MATLABie w postaci funkcji `graytresh`, która dla zadanego obrazu będącego jej argumentem, wyznacza tę wartość. Implementacja jest oparta tzw. metodę Otsu. Funkcja ta zwraca – niezależnie od typu obrazu wejściowego – wartość z zakresu $< 0, 1 >$. Dokonując właściwego progowania należy więc wartość tę – w przypadku obrazów typów innych niż `double` – przekształcić w wartość z zakresu odpowiadającego możliwym wartościom punktów obrazu.

Zadanie 22 *Wykonaj progowanie dowolnego obrazu typu* `uint8` *z automatycznym doborem progu metodą Otsu.*

5.2 Segmentacja koloru

Segmentacja koloru jest odpowiednikiem dla obrazów kolorowych operacji segmentacji przez progowanie. W tym przypadku kryterium przynależności punktu do zadanego obiektu są wartości poszczególnych składowych koloru R,G i B. O ile więc w przypadku progowania obrazu w skali szarości mamy do czynienia z wyznaczeniem przedziału w jednowymiarowej przestrzeni jasności, o tyle w przypadku obrazu kolorowego wyznaczany jest trójwymiarowy obszar w przestrzeni barw odpowiadający kolorom charakterystycznym dla segmentowanego obiektu. Obszar taki może być zdefiniowany na kilka sposobów. Najczęściej określa się w przestrzeni kolorów *punkt odniesienia*, który charakteryzuje barwę, która opisuje te obiekty na obrazie, które zostaną wybrane w wyniki segmentacji. Ponieważ barwa punktów należących do obiektów nie jest jednolita – istnieją w obrębie tych obiektów różnice w barwie punktów, to zakłada się dodatkowo pewien margines zmienności składowych barwy. Dzięki temu punkty o barwie zbliżonej do punktu odniesienia, ale jednak nieco różnej są także poprawnie segmentowane. Określa się więc obszar w przestrzeni kolorów, który opisuje punkty należące do końcowego wyniku segmentacji. Przykład tego typu segmentacji pokazuje Przykład 28.

Wartość tego punktu jest odczytywana bezpośrednio z obrazu przy pomocy instrukcji `impixelinfo`, która powoduje wyświetlenie w oknie obrazu dodatkowo informacji o wartości punktu wskazywanego przez kursor.

Przykład 28 *Segmentacja koloru*

```
>> o = imread('kostka.bmp');
>> o1 = im2double(o);   % konwersja obrazu na typ double
>> imshow(o); impixelinfo;    % wyświetl obraz, pokaz wartości punktów
>> r = 0.95    % wartości przykładowe punktu odniesienia
>> g = 0.88
>> b = 0.22
>> t = 0.3    % margines - próg odległości
>> sr = o1(:,:,1) < r + t & o1(:,:,1) > r - t;
>> sg = o1(:,:,2) < g + t & o1(:,:,2) > g - t;
>> sb = o1(:,:,3) < b + t & o1(:,:,3) > b - t;
```

```
>> segm = sr & sg & sb;
>> imshow(segm);
```

Zadanie 23 *Poeksperymentuj z różnymi wartościami marginesu* `t` *i z różnymi punktami odniesienia* `r,g,b`.

Inny sposób segmentacji wykorzystuje pojęcie odległości między kolorami w przestrzeni kolorów. Mając bowiem zadany punkt odniesienia, możemy wyznaczyć odległość do tego punktu dla każdego punktu obrazu. W ten sposób określamy jak bardzo różnią się poszczególne punkty obrazu od punktu odniesienia. Mając wyliczoną taką odległość można łatwo dokonać segmentacji poprzez progowanie. Punkty, dla których odległość od punktu odniesienia jest mniejsza od zadanego progu otrzymują wartość 1 na obrazie wyjściowym, pozostałe zaś - wartość 0.

Przykład 29 pokazuje w jaki sposób może być wykonana segmentacja obrazu barwnego z wykorzystaniem odległości Euklidesowej do punktu odniesienia typu metryka miejska odczytywanego interaktywnie przy pomocy instrukcji `impixelinfo`. Alternatywnie, kombinację `imshow` oraz `impixelinfo` można zastąpić pojedynczą komendą wyświetlania obrazu `imtool`.

Przykład 29 *Segmentacja koloru – odległość Euklidesowa*

```
>> o = imread('kostka.bmp');
>> o1 = im2double(o);   % konwersja obrazu na typ double
>> imshow(o1); impixelinfo;    % wyświetl obraz, pokaz wartości punktów
>> r = 0.95    % wartości przykładowe
>> g = 0.88
>> b = 0.22
>> [m n n1] = size(o1);
>> rgb = cat(3,r*ones(m,n),g*ones(m,n),b*ones(m,n));
>> imshow(rgb);
>> d1 = (o1 - rgb) .* (o1 - rgb);
>> dist = sqrt ( d1(:,:,1) + d1(:,:,2) + d1(:,:,3));
>> imshow(dist);   % odległości do wybranego koloru
>> p = 0.3   % prog odleglosci
>> segm = dist < p;
>> imshow (segm);
```

Zadanie 24 *Poeksperymentuj z różnymi wartościami progu odległości* `p`.

Trzeci przykład segmentuje obraz wykorzystując do wyznaczania odległości między punktami obrazu, a punktem odniesienia metrykę miejską. Ponadto punkt odniesienia jest w Przykładzie 30 definiowany w inny sposób. Tym razem użytkownik zaznacza na obrazie obszar charakteryzujący się pożądanym zakresem barw. Następnie wyznaczana jest wartość średnia punktów należących do tego obszaru i ta wartość jest przyjmowana jako punkt odniesienia.

Przykład 30 *Segmentacja koloru - ROI + metryka miejska*

```
>> o = imread('kostka.bmp');
>> o1 = im2double(o);   % konwersja obrazu na typ double
>> imshow(o1); impixelinfo;    % wyświetl obraz, pokaz wartości punktów
>> maska = roipoly(o1);    % interaktywne definiowanie obszaru
>> imshow (maska);
>> maska1 = im2double(cat(3, maska, maska, maska));  % konwersja na RGB
>> o2 = immultiply(o1,maska1);    % wybierz tylko obszar zainteresowania
>> imshow (o2);
>> lpunktow = sum(sum(maska))  % liczba punktow maski
```

```
>> r = sum(sum(o2(:,:,1))) / lpunktow   % srednia wartosc R w masce
>> g = sum(sum(o2(:,:,2))) / lpunktow   % srednia wartosc G w masce
>> b = sum(sum(o2(:,:,3))) / lpunktow   % srednia wartosc B w masce
>> [m n n1] = size(o1);
>> rgb = cat(3,r*ones(m,n),g*ones(m,n),b*ones(m,n));
>> imshow(rgb);
>> d1 = abs (o1 - rgb);
>> dist = d1(:,:,1) + d1(:,:,2) + d1(:,:,3);
>> imshow(dist);   % odległości do wybranego koloru
>> p = 0.3   % prog odleglosci
>> segm = dist < p;
>> imshow (segm);
```

Zadanie 25 *Wykonaj segmentację z Przykładu 29 tak, aby uzyskać elementy kostki o innych barwach.*

Zadanie 26 *Napisz funkcję, która dokonuje segmentacji koloru jedną z wybranych metod. Funkcja powinna akceptować cztery parametry wejściowe: zmienną zawierająca kolorowy obraz przeznaczony do segmentacji, trójelementowy wektor zawierający punkt odniesienia, skalarną wartość marginesu/progu odległości, oraz typ segmentacji:* `'szescian'`,`'Euklides'`,`'miejska'`. *Typy segmentacji odpowiadają wariantom omówionym kolejno w przykładach 28, 29, 30. Funkcja powinna zwracać obraz binarny - wynik segmentacji.*

Zadanie 27 *Wykonaj segmentację kolorów analogiczną do pokazanej w przykładzie 28 lecz w przestrzeni HSV. Czy w tym przypadku możliwe jest uzyskanie poprawnych wyników segmentacji wykonując progowanie jedynie jednej lub dwóch składowych ?*

Zadanie 28 *Wykonaj przykład "Color-Based Segmentation Using the L*a*b* Color Space" dostępny w systemie pomocy do Image Processing Toolbox.*

6 Liniowe operacje kontekstowe

Operacje kontekstowe tym różnią się od punktowych (opisanych w p. 3), że na wartość piksela o zadanych współrzędnych na obrazie wyjściowym wpływa nie tylko wartość tego samego piksela na obrazie wejściowym, ale także wartości punktów należących do jego sąsiedztwa. Stanowią one *kontekst* w jakim występuje dany piksel. Spośród metod kontekstowych, najpopularniejszymi są filtry liniowe (splotowe, konwolucyjne), medianowe oraz operatory morfologiczne.

Operacje kontekstowe są najczęściej wykorzystywane do filtracji obrazów. Jednym z celów filtracji jest poprawa jakości obrazów. Uzyskana w wyniku filtracji poprawa jakości może polegać na usunięciu z obrazu szumu, obiektów nie spełniających zadanych kryteriów (np. zbyt małych), na poprawie jego kontrastu, itp. W wyniku filtracji może też powstać obraz zmodyfikowany np. gradientowy.

Filtracja splotowa jest jedną z najczęściej stosowanych metod w przetwarzaniu obrazu. Polega ona na zastosowaniu funkcji splotu na obrazie, co sprowadza się do wyznaczenia - dla każdego punktu obrazu – kombinacji liniowej wartości jego sąsiadów z pewnymi współczynnikami wagowymi. Macierz tych współczynników nazywana jest *maską splotu* i jest stała dla wszystkich punktów obrazu. Maska splotu określa, którzy spośród sąsiadów każdego piksela są uwzględniani i z jakimi wagami. Splot dwuwymiarowy można zdefiniować wzorem:

$$g(x,y) = \sum_{i=-M}^{M} \sum_{j=-N}^{N} h(i+M, j+N) \cdot f(x+i, y+j), \tag{2}$$

przy czym f jest dwuwymiarową funkcją obrazu (obrazem); g jest obrazem po filtracji; h jest maską splotu o rozmiarach $(2M+1) \times (2N+1)$. Współczynniki maski $h(i,j)$ spełniają określone warunki normalizacyjne, które zależą od przeznaczenia danej maski.

Praktyczna realizacja operacji splotu polega na przeglądaniu obrazu punkt po punkcie i na wyznaczaniu wartości punktu na obrazie wyjściowym jako kombinacji liniowej wartości punktów na obrazie wejściowym: punktu o tych samych współrzędnych oraz jego sąsiadów. Rozmiar sąsiedztwa odpowiada rozmiarowi maski filtru, zaś współczynniki kombinacji liniowej są wartościami poszczególnych elementów maski.

6.1 Filtry splotowe dolnoprzepustowe

Filtry splotowe rozmywające są filtrami dolnoprzepustowymi – usuwają składowe o wysokiej częstotliwości tj. punkty o jasności znacząco różnej od punktów je otaczających. W efekcie działania takich filtrów następuje usunięcie szumu z obrazu przy jednoczesnym rozmyciu krawędzi obrazu.

W przypadku filtrów dolnoprzepustowych, maska h spełnia warunek normalizacyjny:

$$\sum_{i=-M}^{M} \sum_{j=-N}^{N} h(i+M, j+N) = 1 \tag{3}$$

Dzięki niemu zbiór wartości funkcji g jest taki sam jak funkcji f.

Poniżej podano kilka przykładów filtrów liniowych dolnoprzepustowych.

- filtry uśredniające

$$h = \frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4}$$

$$h = \frac{1}{5} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{5}$$

- uogólnienie filtru uśredniającego

$$h = \frac{1}{k+8} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & k & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{6}$$

- filtr Gaussa 3x3

$$h = \frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{7}$$

- filtr Gaussa 5x5

$$h = \frac{1}{256} \cdot \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 1 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{8}$$

Instrukcją pozwalającą na liniową filtrację obrazu jest `imfilter(obraz_wej, maska)`, przy czym `maska` jest macierzą prostokątną zawierającą współczynnik maski filtru. Maska filtru może być podana bezpośrednio jako macierz prostokątna lub z wykorzystaniem funkcji `fspecial`, która generuje maski kilku najpopularniejszych filtrów liniowych.

Najczęściej stosuje się następujące filtry rozmywające:

1. Uśredniający

```
maska = fspecial('average', 5)  % filtr uśredniający o masce 5x5
maska = ones(5) / (5*5)  % alternatywna definicja
maska = fspecial('average', 3)  % filtr uśredniający o masce 3x3
maska = ones(3) / (3*3) % alternatywna definicja
maska = [1 1 1 ; 1 1 1 ; 1 1 1] / 9  % alternatywna definicja 2
maska = [0 1 0 ; 1 1 1 ; 0 1 0] / 5  % nieco inna wersja filtru
```

2. Uśredniający uogólniony

```
k = 3  % ta wartość jest definiowana przez użytkownika
maska = [1 1 1 ; 1 k 1 ; 1 1 1] / (k + 8)
```

3. Gaussa

```
maska = [ 1 2 1 ; 2 4 2 ; 1 2 1]  % najprostsza maska 3x3
maska = fspecial('gaussian',3)    % inny rodzaj maski 3x3
maska = fspecial ('gaussian',5)   % maska 5x5
```

Zadanie 29 *Sprawdź jakie parametry mogą zostać przekazane jako parametry filtru Gaussa o masce tworzonej komendą* `fspecial('gaussian',...`.

Przykład 31 *Zastosowanie filtru liniowego rozmywającego*

```
o = imread('uscalszum1.tif');   % wczytaj obraz zaszumiony
m = ones (3)/ (3*3)  %
sum(sum(m))  % czy suma wsp. maski wynosi 1 ?
o1 = imfilter(o,m);
subplot (1,2,1); subimage (o);
subplot (1,2,2); subimage (o1);
```

Zadanie 30 *Wczytaj obraz* `wzortest.tif`. *Korzystając z przykładu 31 oraz odpowiednich wzorów wykonaj operacje rozmycia tego obrazu z wykorzystaniem wszystkich opisanych masek oraz różnych wielkości masek i - jeśli występują - ich parametrów. Porównaj wyniki.*

Pytanie 12 *Jak wartość środkowego elementu (*`k`*) maski w filtrze uśredniającym uogólnionym wpływa na ostateczny wynik filtracji?*

Filtry dolnoprzepustowe wykorzystuje się także do filtracji obrazów kolorowych. W takim przypadku filtrowane są niezależnie wszystkie trzy składowe koloru. Pokazuje to poniższy przykład.

Przykład 32 *Filtr uśredniający obrazu kolorowego*

```
im = imread('peppers.png');
h = ones(5,5)/25;
im2 = imfilter(im,h);
imshow([im ; im2])
```

Zadanie 31 *Zastosuj inne warianty filtru dolnoprzepustowego w przykładzie 32. Porównaj wyniki.*

Zadanie 32 *Wykonaj dowolną segmentację obrazu kolorowego (np. wg. przykładów 28– 30) poprzedzając ją filtracją splotową obrazu początkowego. Jak jej wprowadzenie wpływa na wynik segmentacji ?*

6.2 Filtry splotowe górnoprzepustowe

Filtry górnoprzepustowe usuwają z obrazu jego komponenty o niskiej częstotliwości, pozostawiając te o wysokiej. W praktyce sprowadza się to do usunięcia obszarów o łagodnie zmieniającym się odcieniu szarości (kolorze) a pozostawieniu fragmentów gdzie wartość punktu zmienia się istotnie. Typowymi elementami obrazu tego typu są krawędzie. Dlatego filtry dolnoprzepustowe są najczęściej stosowane do ich wykrywania.

W przypadku filtrów górnoprzepustowych, maska h spełnia inny warunek normalizacyjny:

$$\sum_{i=-M}^{M} \sum_{j=-N}^{N} h(i+M, j+N) = 0 \tag{9}$$

Warunek ten nie gwarantuje jednak tego, że wynik filtracji dla pojedynczego piksela jest zawarty w dopuszczalnym zakresie odcieni szarości obrazu. Dlatego należy w tym przypadku przeprowadzić dodatkową normalizację.

Typowymi filtrami liniowymi wykrywającymi krawędzie są filtry *Prewitta* oraz *Sobela*. Maski tych filtrów uzyskuje się definiują je bezpośrednio, lub korzystając z komend `fspecial('prewitt')` lub odpowiednio `fspecial('sobel')`. Maski filtrów w ten sposób uzyskane pozwalają na wykrywanie krawędzi poziomych. Krawędzie pionowe uzyskuje się stosując transponowane maski filtrów. Obraz zawierający krawędzie obu rodzajów uzyskuje się sumując punktowo oba obrazy, co pokazuje przykład **??**.

Przykład 33 *Wykrywanie krawędzi filtrem Prewitta*

```
im = imread('rose.tif');
im1 = im2double(im);
f = fspecial('prewitt')
im2 = imfilter(im1,f);
max(max(im2))   % maksymalna wartość punktu obrazu przefiltrowanego
min(min(im2))   % i minimalna
im2a = abs(im2/3); % normalizacja
max(max(im2a))   % maksymalna wartość po normalizacji
min(min(im2a))   % i minimalna
im3 = imfilter(im1,f'); % transponowana maska filtru
im3a = abs(im3/3);
im4 = im2a+im3a;
imshow([im1 im2a ; im3a im4]);
```

Pytanie 13 *Na czym polega w tym przypadku normalizacja wyniku filtracji ? Dlaczego normalizując dzielimy przez 3 ?*

Zadanie 33 *Wykonaj analogiczną filtrację filtrem Sobela. Jak należy w tym przypadku normalizować wynik ?*

Pytanie 14 *Jak wyglądają maski filtrów Sobela i Prewitta ?*

Zadanie 34 *Napisz funkcję* `krawedzie(imin,filtr)`*, która wykona filtrację w zależności od parametru* `filtr` *- Sobela, albo Prewitta, obrazu* `imin` *i zwróci jej wynik. Wykorzystaj komendę* `if`*. Informacje o jej składni i sposobie użycia zaczerpnij z systemu pomocy MATLAB-a.*

Filtry górnoprzepustowe wykrywające krawędzie są zaimplementowane w funkcji `edge`, której jednym z parametrów jest rodzaj detektora krawędzi. Oprócz typowych detektorów typu filtry Prewitta i Sobela, funkcja ta umożliwia także wykrywanie krawędzi bardziej zaawansowanymi metodami w tym tzw. detektorem Canny'ego.

Zadanie 35 *Sprawdź jak działa funkcja* `edge` *– jakie jej parametry mogą być zdefiniowane, jakie wyniki można przy jej pomocy uzyskać ?*

Innym filtrem pozwalającym na wykrywanie krawędzi jest Laplasjan. O ile maski Prewitta i Sobel aproksymują pierwszą pochodną obrazu, o tyle Laplasjan to druga jego pochodna. Maska Laplasjanu występuje w wielu wersjach, z których najprostszymi są:

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \; ; \; h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{10}$$

Bardziej zaawansowana maska Laplasjanu, zaimplementowana w MATALB-ie jako `fspecial('laplacian',alfa)` jest następująca:

$$h = \frac{1}{1+\alpha} \cdot \begin{bmatrix} \alpha & 1-\alpha & \alpha \\ 1-\alpha & -4 & 1-\alpha \\ \alpha & 1-\alpha & \alpha \end{bmatrix} \tag{11}$$

Przykład 34 *Laplasjan*

```
im = imread('tool.bmp');
im2 = im2double(im);
f = fspecial('laplacian')
im3 = imfilter(im2,f);
im4 = (im3/6) + 0.5  % z uwzględnieniem znaku drugie pochodnej
im5 = abs(im3/3);  % bez uwzględnienia znaku pochodnej
imshow([im2 im4 im5]);
```

Filtr Laplace'a może być wykorzystany także do poprawy kontrastu obrazu.

Przykład 35 *Zastosowanie Laplasjanu do poprawy kontrastu*

```
im = imread('lenagray.bmp');
im2 = im2double(im);
f = fspecial('laplacian',1);
im3 = imfilter(im2,f);
im4 = im2 - im3;
imshow([im2 im4])
```

Zadanie 36 *Jak wartość parametru α wpływa na wynik poprawy kontrastu obrazu ? Wykonaj stosowne eksperymenty.*

Pytanie 15 *Dlaczego odjęcie Laplasjanu od obrazu powoduje poprawę kontrastu ? Wskazówka: skorzystaj z własności drugiej pochodnej*

Działanie filtrów dolno- i górnoprzepustowych jest w pewnych sytuacjach komplementarne. Pierwsze z nich usuwają z obrazu szum, pozostawiając istotne jego elementy. Usunięcie szumu pozwala z kolei na lepsze wykrycie krawędzi ponieważ nie występują zakłócenia od niego pochodzące.

Zadanie 37 *Wykonaj filtację filtrem Sobela lub Prewitta obrazów* `'rose.tif'` *oraz* `'uscal1.tif''` *po ich uprzedniej filtracji fitrem dolnoprzepustowym. Zaobserwuj różnice w wynikach wykrywania krawędzi z i bez wstępnej filtracji dolnoprzepustowej.*

Zadanie 38 *Napisz funkcję, która dla zadanego obrazu wejściowego, maski filtru splotowego oraz współrzędnej 'x' linii profilu wyznacza wynik filtracji splotowej z zadaną maską oraz profile obrazu*

przed i po filtracji. Profile obrazów powinny być wykonane wzdłuż linii prostopadłej do osi współrzędnych 'x' dla wartości tej współrzędnej podanej jako trzeci parametr wywołania funkcji, zaś zakres zmienności współrzędnej 'y' powinien obejmować cały obraz. Funkcja zwraca obraz wyjściowy – wynik filtracji i dodatkowo wyświetla dwa okna. W pierwszym, z lewej strony pokazany jest obraz wejściowy, zaś z prawej – obraz wyjściowy. W drugim oknie, z lewej strony znajduje się przekrój obrazu wejściowego, zaś z prawej – przekrój obrazu wyjściowego.

Klasycznym połączeniem wspomnianego typu jest połączenie filtru Gaussa i Laplasjanu. Jest ono nazywane filtrem LoG (z ang. „Laplacian of Gaussian").

Przykład 36 *Złożenie Laplasjanu i filtru Gaussa*

```
im = imread('lenagray.bmp');
im2 = im2double(im);
f = fspecial('laplacian');
im3 = imfilter(im2,f);
im4 = abs(im3/3);    % zwykły Laplasjan
imshow([im2 im4*8])
im12 = imfilter(im2,fspecial('gaussian',[5 5],0.7)); % filtr Gaussa
im13 = imfilter(im12,f);
im14 = abs(im13/3);   % Laplasjan po filtrze Gaussa
imshow([im2 im4*8 ; im12 im14*8])
```

Pytanie 16 *Jaki efekt pozytywny uzyskaliśmy dzięki zastosowaniu filtru Gaussa przed wyznaczeniem Laplasjanu ?*

7 Nieliniowe operacje kontekstowe

W przypadku liniowych operacji kontekstowych, zależność wartości piksela wyjściowego od odpowiednich pikseli obrazu wejściowego miała charakter liniowy. Zależność ta może być także nieliniowa. Najczęściej polega ona na przypisaniu pikselowi obrazu wyjściowego wartości jednego, wybranego wśród sąsiadów, piksela na obrazie wejściowym.

7.1 Filtr medianowy

W przeciwieństwie do filtrów opisanych w poprzednich punktach, *filtr medianowy* nie jest filtrem liniowym tj. wartość punktu na obrazie wyjściowym nie jest liniowo zależna od wartości punktów na obrazie wejściowym. W przypadku filtru medianowego wyjściową wartością punktu jest mediana wartości punktów sąsiedztwa.

Dla dowolnego zbioru liczb *medianę* definiujemy jako wartość środkową wyznaczoną po uprzednim uporządkowaniu rosnąco (lub malejąco) tego zbioru.

Przykład 37 *Mediana*

```
w = [2 9 3 2 1 0 10 33 5]   % dowolny wektor
median(w)    % wartość mediany
sort(w)      % uporządkowany wektor, gdzie jest mediana?
```

Działanie obrazowego filtru medianowego jest podobne do działania filtru liniowego. Jednak zamiast wyznaczania dla każdego punktu obrazu kombinacji liniowej, wyznaczana jest mediana spośród punktów sąsiedztwa aktualnie przetwarzanego punktu obrazu.

Przykład 38 *Zastosowanie filtru medianowego*

```
o = imread('uscalszum1.tif');   % wczytaj obraz zaszumiony
o1 = medfilt2 (o, [3 3]);    % filtr medianowy z maską 3x3
subplot (1,2,1); subimage (o);
subplot (1,2,2); subimage (o1);
```

Zadanie 39 *Wykonaj filtrację medianową obrazu* `uscalszum1.tif` *dla różnych rozmiarów masek. Porównaj wyniki.*

Pytanie 17 *Czym różni się efekt filtracji liniowej i medianowej? Który filtr lepiej usuwa szum?*

Zadanie 40 *Napisz funkcję, która dla zadanego obrazu wejściowego, maski filtru medianowego oraz parametru gamma wykonuje korekcję gamma obrazu wejściowego, a następnie na jej wyniku filtrację medianową z zadaną maską. Ponadto funkcja wyznacza histogramy obrazu przed i po filtracji. Funkcja zwraca obraz wyjściowy – wynik filtracji i dodatkowo wyświetla dwa okna. W pierwszym, na górze pokazany jest obraz wejściowy, zaś na dole – obraz wyjściowy. W drugim oknie, na górze znajduje się histogram obrazu wejściowego, zaś na dole – histogram obrazu wyjściowego.*

7.2 Podstawy morfologicznego przetwarzania obrazów

Morfologiczne metody przetwarzania obrazów oparte są o nieliniowe operatory działające w sąsiedztwie punktu. Są to operatory minimum i maksimum. W oparciu o nie definiuje się dwie podstawowe operacje morfologiczne: *erozję* i *dylację*.

Otoczenie punktu jest w morfologii matematycznej definiowane jako *element strukturujący*. Jest on odpowiednikiem maski filtru liniowego i określa obszar sąsiedztwa, w jakim wyznaczana jest wartość minimalna bądź maksymalna. W przeciwieństwie do maski, w najprostszej postaci elementu strukturującego punkty do niego należące nie mają wartości liczbowych. Element strukturujący jest macierzą dwuwymiarową, w której elementy o wartości 1 odpowiadają punktom otoczenia należącym do elementu strukturującego, punkty zaś o wartości 0 – do niego nie należącymi. Przykładowymi elementami strukturującymi są:

- Element jednostkowy 4 - spójny:
  `se = [0 1 0 ; 1 1 1 ; 0 1 0]`
- Element jednostkowy 8 - spójny:
  `se = ones(3)`
- Element kierunkowy pionowy:
  `se = [0 1 0 ; 0 1 0 ; 0 1 0]`
- Element kierunkowy poziomy:
  `se = [0 0 0 ; 1 1 1 ; 0 0 0]`
- Elementy kierunkowe diagonalne:
  `se = [1 0 0 ; 0 1 0 ; 0 0 1]`
  `se = [0 0 1 ; 0 1 0 ; 1 0 0]`

W celu uzyskania elementu strukturującego o większych rozmiarach należy użyć instrukcji `strel`, która zwraca element strukturujący zadanego typu. Do najczęściej wykorzystywanych typów należą:

- *Kulisty* obejmujący punkty należące do koła o zadanym promieniu:
  `>> se = strel('disk',R)` gdzie `R` jest promieniem koła.

- *Kwadratowy* będący kwadratem o zadanych rozmiarach:
  `>> se = strel('square', B)` gdzie `B` jest długością boku kwadratu.
- *Romboidalny* o kształcie rombu o równych przekątnych:
  `>> se = strel('diamond',D)` gdzie `D` jest połową długości przekątnej.
- *Liniowy* będący linią prostą pod zadanym kątem i zadanej długości:
  `>> se = strel('line',D,K)` gdzie `D` jest długością, zaś `K` - kątem.

Zadanie 41 *Zobacz jak wyglądają elementy strukturujące uzyskiwane dzięki instrukcji* `strel` *dla różnych wartości argumentów. Jak przy dobierając odpowiednie argumenty instrukcji* `strel` *uzyskać element strukturujący jednostkowy 4- i 8-spójny?*

7.3 Erozja i dylacja

Erozja polega na wyznaczeniu dla każdego punktu obrazu wyjściowego wartości minimalnej spośród punktów otoczenia opisanych przy pomocy elementu strukturującego na obrazie wejściowym. Dualna do niej operacja *dylacji* polega na wyznaczaniu odpowiednio wartości maksymalnej. Erozja i dylacja mogą być wyznaczane dla obrazów binarnych i w skali szarościch.

Dylacja[2] i *erozja* obrazu f elementem strukturującym B są definiowane odpowiednio jako:

$$\begin{aligned} \delta_B(f(p)) &= \sup_{r\in B}[f(p+r)] \\ \varepsilon_B(f(p)) &= \inf_{r\in B}[f(p+r)] \end{aligned} \tag{12}$$

Są to więc operatory przypisujące każdemu punktowi obrazu wartość odpowiednio najmniejsza lub najwiekszą spośród punktów sąsiedztwa określonego przez element strukturujący B.

Komendami pozwalającymi na wykonanie tych operacji są odpowiednio `imerode` oraz `imdilate`, których argumentami są obraz oraz element strukturujący.

Przykład 39 *Erozja i dylacja*

```
>> o = imread('uscal.tif');
>> figure('Name','Obraz poczatkowy');imshow(o);
>> se1 = [0 1 0 ; 1 1 1 ; 0 1 0]     % pierwszy element strukturujący
>> se2 = strel('disk',5)  % drugi element strukturujący
>> oe1 = imerode(o, se1);
>> figure('Name','Wynik erozji z se1');imshow(oe1);
>> oe2 = imerode(o, se2);
>> figure('Name','Wynik erozji z se2');imshow(oe2);
>> od1 = imdilate(o, se1);
>> figure('Name','Wynik dylacji z se1');imshow(od1);
>> od2 = imdilate(o, se2);
>> figure('Name','Wynik dylacji z se2');imshow(od2);
```

Zadanie 42 *Wykonaj eksperymenty z erozją i dylacją dla różnych elementów strukturujących, o różnych kształtach i wielkościach na obrazach: w skali szarościm* `wzortest.tif` *oraz binarnym* `wzorbin.tif`*. W tym ostatnim przypadku wykonaj też operacje z elementami strukturującymi kierunkowymi.*

Zadanie 43 *Dla obrazu binarnego* `sets4.bmp` *przeprowadź operacje erozji i dylacji kierunkowej w różnych kierunkach. Wykonaj te same eksperymenty dla negatywu obrazu* `sets4`*.*

[2] W literaturze polskojęzycznej nazywana też dylatacją.

7.4 Gradient morfologiczny

Z faktu, że dla obrazów w odcieniach szarości erozja niejako „rozszerza" ciemniejsze fragmenty obrazu, „zwężając" jaśniejsze wynika, że odejmując wynik erozji od obrazu początkowego otrzymamy obraz zawierający jedynie zmiany „nachylenia" funkcji obrazu. Funkcja taka jest gradientem funkcji obrazu, a ściśle gradientem przez erozję. Gradient przez dylację definiuje się jako różnicę wyniku dylacji i obrazu początkowego. Z kolei odejmując wynik erozji od wyniku dylacji otrzymamy *gradient morfologiczny*. Gradienty obrazów binarnych nazywa się też funkcjami konturu ponieważ ich wynikiem jest kontur zbioru.

Powyższe gradienty definiuje się w sposób następujący:

$$\begin{aligned} g_e(f) &= f - \varepsilon(f) \\ g_d(f) &= \delta(f) - f \\ g(f) &= \delta(f) - \varepsilon(f) \end{aligned} \tag{13}$$

Gradienty wyznacza się stosując elementy strukturujące jednostkowe 4- i 8-spójne lub kierunkowe.

Zadanie 44 *Napisz trzy funkcje realizujące omówione typy gradientów (przez erozję, przez dylację i morfologiczny) korzystając z komend* `imerode`, `imdilate` *oraz* `imsubtract`. *Argumentami wejściowym powinny być obrazy dla których wyznaczamy gradienty oraz element strukturujący. Funkcje powinny zwracać obrazy przedstawiające odpowiednie gradienty.*

Zadanie 45 *Zapisz wszystkie trzy funkcje z przykładu 44 jako jedną funkcję o trzech argumentach. Dwa pierwsze są identyczne z argumentami funkcji z wcześniejszego przykładu. Trzeci argument określa rodzaj gradientu może przyjmować jedną z trzech wartości: 'ero','dil','morf'. W zależności od tego parametru funkcja zwraca odpowiedni typ gradientu. Do sprawdzenia wartości trzeciego argumentu wykorzystaj funkcję wbudowaną* `strcmp`, *sposób użycia sprawdź z systemie pomocy.*

Zadanie 46 *Wyznacz gradienty obrazów w skali szarości* `cells1.bmp`, `cells3.bmp`, `chaos3.bmp`, `tool.bmp`, `tool2.bmp` *oraz binarnych* `grains1.bmp` *oraz* `grains2.bmp` .

Pytanie 18 *Jaką grubość ma linia konturu obiektu na obrazie binarnym wyznaczonego metodą gradientu przez erozję, przez dylację oraz gradientu morfologicznego z jednostkowymi elementami strukturującymi ?*

Zadanie 47 *Posegmentowany obraz wynikowy z przykładu 27 zawiera liczne drobne punkty, które nie powinny należeć do poprawnego wyniku segmentacji. Dobierz filtr, który – zastosowany przed segmentacją – pozwoli na poprawę jakości obrazu posegmentowanego. Czy taki filtr można też zastosować po wykonaniu segmentacji, na obrazie posegmentowanym?*

7.5 Otwarcie i zamknięcie

Poprzez złożenie erozji i dylacji otrzymuje się dwa podstawowe *filtry morfologiczne*: otwarcie i zamknięcie. *Otwarcie* definiuje się jako złożenie erozji i dylacji w ten sposób, że w pierwszej kolejności wykonywana jest erozja obrazu wejściowego, a następnie na wyniku erozji wykonywana jest dylacja. W przypadku *zamknięcia*[3] kolejność operacji jest odwrotne. Dzięki takiej kombinacji otwarcie pozwala na usuwanie szumu typu „sól" składającego się z punktów jaśniejszych od tła, zaś zamknięcie umożliwia usunięcie szumu typu „pieprz", czyli drobnych punktów od tła ciemniejszych. Formalnie, operacje te definiuje się następująco - kolejno otwarcie i zamknięcie:

[3] Nazywanego też *domknięciem*

$$\begin{aligned} \gamma_B(f) &= \delta_B \varepsilon_{B^T}(f) \\ \phi_B(f) &= \varepsilon_B \delta_{B^T}(f) \end{aligned} \tag{14}$$

gdzie B jest elementem strukturującym, a B^T – jego transpozycją ($B^T = \{p : -p \in B\}$).

Otwarcie jest wykonywane w MATLAB-ie przy pomocy komendy `imopen`, zaś zamknięcie – `imclose`.

Usunięcie szumu o zróżnicowanym charakterze uzyskujemy przez naprzemienne stosowanie filtrów otwarcia i zamknięcia (lub odwrotnie). Takie złożenie filtrów nazywamy *morfologicznym filtrem przemiennym*.

Przykład 40 *Otwarcie i zamknięcie*

```
>> o1 = imread('uscalszum4.tif');  % filtr typu otwarcie
>> se = strel('diamond',1)
>> otw = imopen(o1,se);
>> figure('Name','filtracja otwarciem');
>> subplot(1,2,1); subimage(o1);
>> subplot(1,2,2); subimage(otw);
>> o2 = imread('uscalszum3.tif');
>> zam = imclose(o2,se);    % filtr typu zamknięcie
>> figure('Name','filtracja zamknieciem');
>> subplot(1,2,1); subimage(o2);
>> subplot(1,2,2); subimage(zam);
>> o3 = imread('uscalszum1.tif');  % filtr przemienny
>> se1 = strel('diamond',2);
>> prz = imclose(imopen(o3,se),se1);
>> figure('Name','filtracja otwarciem i zamknieciem');
>> subplot(1,2,1); subimage(o3);
>> subplot(1,2,2); subimage(prz);
```

Zadanie 48 *Wykonaj eksperymenty z otwarciem i zamknięciem dla różnych elementów strukturujących, o różnych kształtach i wielkościach na obrazach: w skali szarościm* `wzortest.tif` *oraz binarnym* `wzorbin.tif`*. W tym ostatnim przypadku wykonaj też operacje z elementami strukturującymi kierunkowymi.*

Zadanie 49 *Sprawdź na dowolnym obrazie czy wynik operacji* `imopen` *oraz* `imclose` *to rzeczywiście odpowiednie sekwencyjne złożenia erozji i dylacji. Wykorzystaj w tym celu komendy* `imerode` *oraz* `imdilate`[4]*.*

Zadanie 50 *Wykonaj filtrację obrazu* `sets5.bmp` *przy pomocy otwarcia i zamknięcia. Która z tych operacji jest lepsza do odszumienia tego konkretnego obrazu ?*

Zadanie 51 *Na obrazie* `lines3.bmp` *wykonaj otwarcie i zamknięcie z kierunkowymi elementami strukturującymi. Która z tych operacji lepiej wykrywa linie w danym kierunku ? Którą operację należałoby wykonywać dla tego samego obrazu, ale w negatywie ?*

Pytanie 19 *Porównaj filtrację obrazów w skali szarości poznanymi dotychczas filtrami liniowymi, medianowym i morfologicznymi? Czym różnią się wyniki?*

[4]Zadanie to wymaga sprawdzenia czy dwa obrazy są identyczne. Odpowiedni test można wykonać na wiele sposobów. Jeden z nich polega na wykorzystaniu komend: `max(max(im1 == im2))` lub `(max(max(imabsdiff(im1,im2))))`. Podwójna funkcja `max(max(..))` zwraca maksymalną wartość spośród elementów macierzy dwuwymiarowej, będącej jej argumentem.

Przydatnymi z praktycznego punktu widzenia operacjami morfologicznymi są operacje detekcji szczytów i dolin cylindryczny. Pozwalają one na usuwanie tła obrazu z pozostawieniem obiektów znajdujących się na nim. Wyróżnia się dwa rodzaje tego rodzaju operacji: detekcja szczytów (ang. white Top-Hat) oraz detekcja dolin (ang. black Top-Hat). Pierwsza z nich pozostawia obiekty jaśniejsze od tła na obrazie, druga natomiast - ciemniejsze. Definiuje się je w sposób następujący (kolejno biała i czarna transf. cylindryczna):

$$\begin{aligned} WTH_B(f) &= f - \gamma_B(f) \\ BTH_B(f) &= \phi_B(f) - f \end{aligned} \tag{15}$$

Zadanie 52 *Napisz dwie funkcje realizujące obie wersje operatora detekcji szczytów i dolin korzystając z komend* `imopen`, `imclose` *oraz* `imsubtract`. *Argumentami wejściowym powinien być obraz początkowy oraz element strukturujący. Funkcje powinny zwracać wyniki odpowiednich operatorów cylindrycznych.*

Zadanie 53 *Wykonaj operacje detekcji szczytów i dolin na obrazach* `tool`, `tool2`. *Zastosuj element strukturujący typu* `strel('square',n)` *gdzie* `n` *jest rozmiarem elementu. Jaki rozmiar elementu strukturującego daje najlepsze wyniki ? Przez najlepsze jest tu rozumiana najlepsza ekstrakcja obiektów z tła. Która z wersji operacji (BTH, WTH) odfiltrowuje tło ciemniejsze od obiektów, a która jaśniejsze ?*

Element strukturujący otwarcia/zamknięcia jest zależny od wielkości obiektów, które mają pozostać na obrazie.

7.6 Morfologiczne filtry rekonstrukcyjne

Operacja rekonstrukcji jest jedną z ciekawszych i bardziej użytecznych operacji morfologicznych. Pozwala ona na odtwarzanie pewnych elementów obrazu na podstawie ich fragmentów. Dzięki temu jest również możliwe usuwanie niepotrzebnych części obrazu, bez jednoczesnego pogarszania jakości pozostałej jego części.

Rekonstrukcja przez dylację obrazu-maski g z obrazu ze znacznikami f ($f \leq g$) jest definiowana jako dylacja geodezyjna o rozmiarze jednostkowym obrazu f z maska g wykonywana iteracyjnie aż do spełnienia warunku idempotentności (czyli wykonywana do chwili gdy obraz nie ulegnie zmianie po kolejnej jednostkowej dylacji geodezyjnej). Oznacza się ją następująco:

$$R_g(f) = \delta_g^{(i)}(f), \tag{16}$$

przy czym i jest takie, że:

$$\delta_g^{(i)}(f) = \delta_g^{(i+1)}(f) \tag{17}$$

Rekonstrukcja przez erozję obrazu-maski g z obrazu ze znacznikami f ($f \geq g$) jest definiowana jako erozja geodezyjna jednostkowa obrazu f z maską g wykonywana aż do spełnienia warunku idempotentności (czyli do chwili gdy obraz nie ulegnie zmianie po kolejnej jednostkowej erozji geodezyjnej). Oznacza się ją jako:

$$R_g^*(f) = \varepsilon_g^{(i)}(f) \tag{18}$$

przy czym i jest takie, że:

$$\varepsilon_g^{(i)}(f) = \varepsilon_g^{(i+1)}(f) \tag{19}$$

Rekonstrukcję przez erozję oraz rekonstrukcję przez dylację wiąże następująca zależność:

$$R_g^*(f) = \overline{R_{\overline{g}}(\overline{f})}, \tag{20}$$

przy czym $\overline{f}$ i $\overline{g}$ oznaczają negatywy obrazów odpowiednio f i g.

Rekonstrukcja morfologiczna pozwala na usuwanie zbędnych fragmentów obrazu przy jednoczesnym pozostawieniu w niezmienionej formie pozostałej jego części. Przy pomocy znaczników wskazuje się te elementy, które powinny pozostać na obrazie. Po dokonaniu rekonstrukcji znaczników z maską równą obrazowi początkowemu wszystkie obszary obrazu, które nie zostały zaznaczone znikają.

Przykład 41 *Ekstrakcja wybranych elementów obrazu binarnego*

```
>> im = imread('grains1.bmp');
>> im1 = imcomplement(im);
>> imshow(im1)
>> im2 = zeros(size(im1,1),size(im1,2));
>> im2(100,100)=1;
>> im2(5,40)=1;
>> im2(85,230)=1;
>> im2(220,100)=1;
>> im2a=im2uint8(im2);
>> im3 = imreconstruct(im2a,im1);
>> imshow([im1 im2a im3]);
```

Zaznaczanie poszczególnych elementów obrazu może też być wykonywane ręcznie przy pomocy myszki. Do interaktywnego wyznaczania obszaru zainteresowania (ang. „region of interest" - ROI) służy instrukcja `roipoly`, która wyświetla obraz i oczekuje od użytkownika oznaczenia przy pomocy krzywej łamanej obszaru. Pojedyncze kliknięcie wprowadza kolejny punkt krzywej, podwójne zaś kliknięcie - kończy wprowadzanie obszaru.

Przykład 42 *Ekstrakcja wybranych interaktywnie elementów obrazu binarnego*

```
>> im = imread('grains1.bmp');
>> im1 = imcomplement(im);
>> imshow(im1)
>> p = roipoly;
>> p2 = min(im2uint8(p),im1);
>> im2 = imreconstruct(p2,im1);
>> imshow([im1 p2 im2]);
```

W analogiczny sposób można wybierać obiekty z obrazów w skali szarości.

Zadanie 54 *Wykonaj przykład 42 dla obrazu* `kalkulator.tif` *(bez wyznaczania jego negacji), zaznaczając przy pomocy myszki napisy na klawiszach kalkulatora znajdującego się na zdjęciu.*

Umiejętnie dobierając znaczniki można znajdować obiekty znajdujące się w konkretnych miejscach obrazu. Jeżeli więc obraz ze znacznikami będzie się składał z ramki obrazu o szerokości 1 piksela to w wyniku rekonstrukcji takiego obrazu z maską równą obrazowi początkowemu otrzymamy jedynie te fragmenty obrazu, które dotykają do krawędzi obrazu. Odejmując następnie ten obraz od obrazu początkowego otrzymamy jedynie te obiekty, które w całości mieszczą się na obrazie.

Przykład 43 *Ekstrakcja i usuwanie elementów dotykających brzegu obrazu*

```
>> im = imread('grains1.bmp');
>> b = imread('border.bmp');
>> im1 = imcomplement(im);
>> im2 = imreconstruct(min(b,im1),im1);
>> im3 = imabsdiff(im1,im2);
>> imshow([im1 im2 im3])
```

Operacja rekonstrukcji może być także stosowana do likwidowania minimów obrazu nie dotykających jego brzegów. Nazywana jest ona wówczas także operacją usuwania otworów (gdyż takie właśnie minima są w pewnym sensie „otworami" w obrazie). Jeżeli obraz początkowy zostanie odwrócony, a następnie zrekonstruowany ze znacznikami (jednopikselową ramką obrazu), to w wyniku otrzymamy tylko tę część tła obrazu początkowego, która dotyka do krawędzi obrazu. Fragmenty tła, które nie dotykają do brzegu znikną. Jeżeli teraz obraz ten zostanie odwrócony, to otrzymamy w wyniku obraz identyczny z początkowym z tą tylko różnica, że wszystkie „otwory" w znajdujących się na nim obiektach zostaną usunięte.

Przykład 44 *Usuwanie otworów*

```
>> im = imread('grains5.bmp');
>> b = imread('border.bmp');
>> im1 = imreconstruct(min(b,im),im);
>> imshow([im im1])
```

Rekonstrukcja morfologiczna jest operacją składową efektywnych filtrów morfologicznych: *otwarcia przez rekonstrukcję* oraz *zamknięcia przez rekonstrukcję*.

Definiuje się je analogicznie do tradycyjnego otwarcia i zamknięcia. Nie są one jednak prostymi złożeniami erozji i dylacji, lecz złożeniami wykorzystującymi operacje rekonstrukcji obu typów. I tak, otwarcie przez rekonstrukcję składa się z erozji o danym rozmiarze oraz wykonanej na jej wyniku rekonstrukcji (przez dylację). Zamknięcie przez rekonstrukcję składa się natomiast z dylacji o zadanym rozmiarze oraz wykonanej na jej wyniku rekonstrukcji dualnej (przez erozję). Maską w obu przypadkach jest obraz początkowy. Formalnie oba omawiane operatory definiuje się następująco:

$$\begin{aligned} \widetilde{\gamma}_B(f) &= R_f(\varepsilon_B(f)) \\ \widetilde{\phi}_B(f) &= R_f^*(\delta_B(f)) \end{aligned} \tag{21}$$

przy czym f jest obrazem początkowym.

W porównaniu z klasycznym otwarciem i zamknięciem opisane powyżej operacje znacznie poprawiają wyniki filtracji. Oprócz bowiem pozytywnych efektów takich jak usunięcie ciemniejszych lub jaśniejszych składników obrazu otwarcie i zamknięcie powoduje także efekty negatywne. Polegają one na zniekształceniu konturów obiektów znajdujących się na obrazie. Otwarcie i zamknięcie przez rekonstrukcje są pozbawione tej wady - usuwają zbędne składniki obrazu bez zniekształcania konturów. Dzięki temu nadają się nie tylko do filtracji obrazów stosowanej jako samodzielna operacja, ale również do segmentacji obrazu (przy wykorzystaniu dodatkowo klasteryzacji). Operacje te wykorzystuje się rownież z powodzeniem w filtrach przemiennych.

Przykład 45 *Porównanie otwarcia klasycznego i przez rekonstrukcję*

```
>> im = imread('grains2.bmp');
>> im1 = imcomplement(im);
>> s = strel('square',11)
>> im2 = imerode(im1,s);
>> im3 = imdilate(im2,s);
>> im4 = imreconstruct(im2,im1);
>> imshow([im1 im2;im3 im4])
```

Zadanie 55 *Napisz funkcję realizującą otwarcie przez rekonstrukcję.*

Zadanie 56 *Porównaj wynik klasycznego otwarcia i otwarcia przez rekonstrukcję dla dowolnych obrazów zawierających jasne obiekty na ciemniejszym tle.*

Zadanie 57 *Rekonstrukcja zaimplementowana w MATLAB-ie jest to rekonstrukcja przez dylację. Napisz funkcję realizującą zamknięcie przez rekonstrukcję wykorzystującą zależność 20.*

Zadanie 58 *Porównaj wynik klasycznego zamknięcia i zamknięcia przez rekonstrukcję dla dowolnych obrazów zawierających ciemne obiekty na jaśniejszym tle.*

Zadanie 59 *Podobnie do tradycyjnych operacji detekcji szczytów i dolin, definiuje się operacje rekonstrukcyjnej detekcji szczytów/dolin. W ich przypadku zamiast klasycznego otwarcia/zamknięcia stosuje się otwarcie lub zamkniecie przez rekonstrukcję. Napisz funkcję realizująca te operacje (analogicznie do funkcji z zadania 52. Porównaj tradycyjne operacje detekcji szczytów i dolin z rekonstrukcyjnymi. Wykorzystaj zadaniu celu obrazy* `tool` *oraz* `tool2`*.*

7.7 Segmentacja morfologiczna

Segmentacja morfologiczna jest sposobem segmentacji obrazu wykorzystującym koncepcję linii działu wodnego (wododziału). Pojęcie to zaczerpnięte zostało z geografii gdzie opisuje linię dzielącą teren na pewne obszary – zlewiska zbiorników wodnych. W przetwarzaniu obrazów, linia działu wodnego (LDW) znalazła zastosowanie do segmentacji, gdyż pozwala na podzielenie obrazu na obszary odpowiadające obiektom znajdującym się na obrazie.

Kluczowe w idei LDW jest pojęcie zlewiska. W przypadku obrazu za zlewisko przyjmuje się obszar, którego centrum jest zawsze lokalne minimum obrazu. Patrząc na obraz jak na fragment terenu, można sobie wyobrazić, że kropla wody, która spadnie na dowolny jego punkt, spłynie do najbliższego „dołka", czyli lokalnego minimum. Granicę obszaru „przyciągania" kropli przez lokalne minimum wyznacza właśnie LDW.

Do wyznaczenia segmentacji wododziałowej służy komenda `watershed`, której wykorzystanie pokazuje przykład 46. Obraz wynikowy jest obrazem z etykietami. Etykieta piksela określa zlewisko do którego dany piksel należy. Obszary o etykiecie równej 0 odpowiadają liniom wododziału, rozdzielającym zlewiska.

Przykład 46 *Linia wododziału*

```
>> im = zeros(100,100);
>> im(30:60,30:60)=ones(31,31);
>> im(34:56,34:56)=zeros(23,23);
>> im2 = im2uint8(im);
>> w = watershed(im2);
>> imshow([im w*100]); % dla lepszej widoczności etykiety mnzymy przez 100
```

Kluczowym aspektem segmentacji wododziałowej jest odpowiednie przekształcenie obrazu przed wykonaniem operacji wyznaczenia LDW. Istotne jest takie jego przefiltrowanie, by w efekcie obraz gradientowy zawierał jedynie istotne minima lokalne odpowiadające tym obiektom, które powinny znaleźć się na obrazie posegmentowanym. Zbyt wiele minimów lokalnych obrazu prowadzi bowiem do jego *przesegmentowania*, tj.takiego wyniku segmentacji, w którym występuje zbyt wiele obszarów nieodpowiadającym rzeczywistym obiektom na obrazie. Do wykrywania minimów regionalnych służy komenda `imregionalmin`. Zjawisko przesegmentowania pokazuje przykład 47.

Przykład 47 *Linia wododziału – zbyt dużo minimów*

```
>> im = imread('chaos2.bmp');
>> imshow(im);
>> w = watershed(im);
>> figure; imshow(w,rand(max(max(w)),3));   % zbyt dużo obszarów
>> m = imregionalmin(im); % minima obrazu
>> figure; imshow(m*0.5 + (w==0)); % a tu widać przyczynę
```

Pytanie 20 *Jak powstał obraz wyświetlany w ostatniej linii przykładu 47 ? Co jest na nim uwidocznione ?*

Zadanie 60 *Wykonaj ten sam eksperyment dla obrazu* `'chaos3.bmp'`*. Zwróć uwagę na przesegmentowanie i jego przyczynę .*

Istotną w tym kontekście operacją jest operacja *wymuszania minimów*. Operacja ta modyfikuje minima znajdujące się na obrazie, usuwając wszystkie minima niepotrzebne, wymuszając jednocześnie powstanie tych potrzebnych. Lokalizacja minimów jest określana w dodatkowym obrazie – masce, gdzie wartości 1 są przypisywane punktom należącym do żądanych minimów, zaś wartości 0 – nienależącym. Wymuszanie minimów jest operacją wykorzystującą rekonstrukcję morfologiczną. Komendą realizującą wymuszanie minimów jest `imimposemin`, której pierwszym argumentem jest obraz na którym zostaną wymuszone minima, zaś drugim - maska wskazująca na żądane minima.

Przykład 48 *Linia wododziału – zbyt dużo minimów*

```
>> im = imread('chaos2.bmp');
>> w = watershed(im);
>> m = imregionalmin(im);
>> t = (im < 10);
>> imshow([m t]); % który obraz jest lepszy ?
>> im2 = imimposemin(im,t);
>> w1 = watershed(im2);
>> m1 = imregionalmin(im2);
>> figure; imshow([w==0 w1==0; m m1]); % stary i nowy wynik
>> figure;
>> imshow(min(im,im2uint8(w1>0))); % inna wizualizacja
```

Zadanie 61 *Dokonaj w podobny sposób prawidłowej segmentacji (tj. bez przesegmentowania) obrazu* `'chaos3.bmp'`*.*

Pytanie 21 *Zaproponuj inne sposoby wyznaczania maski dla operacji wymuszania minimów.*

Linia działu wodnego z definicji przebiega „grzbietami" otaczającymi dane zlewisko, oddzielając je od innych sąsiadujących z nim obszarów zlewiskowych. Obraz będący wynikiem operacji LDW jest obrazem binarnym, zawierającym jedynie granice pomiędzy poszczególnymi zlewiskami. W przypadku obrazów zawierających obiekty znajdujących się jednolitym tle należy uprzednio tak przekształcić obraz, by każdy obiekt znajdujący się na nim był otoczony takim właśnie „grzbietem". Tylko wówczas bowiem będzie mógł powstać prawidłowy wynik segmentacji – linia otaczająca dany obiekt. Oczywistym przekształceniem, jakie stosuje się w tym celu, jest wyznaczenie gradientu obrazu, na którym wyznaczana jest LDW.

Przykład 49 *Linia wododziału*

```
>> im = ones(100,100);
>> im(10:30,10:30)=zeros(21,21);
>> im(40:60,10:30)=zeros(21,21);
>> im(50:90,50:70)=zeros(41,21);
>> imshow(im)
>> im2 = im2uint8(im);
>> im3 = imfilter(im2,fspecial('average',3),'replicate');
>> im4 = im3 - imerode(im3,ones(3,3)); % gradient przez erozję
>> figure; imshow([im2 im3 im4]);
>> w1 = watershed(im4);
>> imshow(w1>0); % linia wododziału na gradiencie
>> w2 = watershed(im3);
>> figure;
>> imshow(w2>0); % dla porównania LDW na obrazie org.
```

Pytanie 22 *Porównaj wyniki segmentacji wododziałowej uzyskanej bez uprzedniego wyznaczenia gradientu i wykonanej na obrazie gradientowym ? Skąd biorą się różnice ?*

Zadanie 62 *Segmentacja metodą LDW daje w wyniku segmentację konturową tj. obraz binarny zawierający kontury obiektów znajdujących się na obrazie początkowym. Inny typ wyniku segmentacji to segmentacja obszarowa, gdzie na obrazie wynikowym wszystkie punkty należące do obiektu mają tą samą wartość, np. 1, zaś nienależące - 0. Segmentację obszarową uzyskujemy w wyniku np. progowania. Jaką operację należy zastosować, by przekształcić wynik segmentacji LDW w wynik typu obszarowego, tak by wszystkie punkty należące do obiektów, a nie tylko ich kontury charakteryzowały się wartością 1 ? Wykonaj odpowiednie eksperymenty.*

Zadanie 63 *Wykonaj przykład "Marker-Controlled Watershed Segmentation" dostępny w systemie pomocy do Image Processing Toolbox.*

8 Wyznaczanie cech obrazu

8.1 Etykietowanie

W wyniku segmentacji powstaje najczęściej obraz binarny zawierający pewną liczbę obiektów, które zwykle w kolejnych krokach poddane zostają analizie. Analiza taka polega na kolejnym przetworzeniu tych obiektów. Dlatego istotne jest wyodrębnienie z obrazu binarnego poszczególnych obiektów. Do tego celu służy operacja *etykietowania*. Polega ona na przetworzeniu obrazu binarnego na obraz, w którym każdy punkt zawiera etykietę określającą jego przynależność do poszczególnych obiektów. Obiekty na obrazie odpowiadają składowym spójnym zbioru pikseli o wartości 1. Do etykietowania wykorzystuje się w MATLAB-ie instrukcję `bwlabel`. Jej zastosowanie pokazuje Przykład 50.

Przykład 50 *Etykietowanie*

```
>> o = imread('dowels.tif');
>> o1 = medfilt2(o,[5 5]);  % filtracja
>> t = graythresh(o1)
>> seg = im2bw(o1,t);   % segmentacja przez progowanie
>> [e, num] = bwlabel(seg);  % etykietowanie
>> num   % wyswietl liczbe etykiet
>> imshow(e); impixelinfo; % sprawdzić wartości etykiet najeżdżając na obiekty!
>> imshow(e+1,rand(num,3));   % lepsza wizualizacja - kazda etykieta to inny kolor
```

Instrukcja `bwlabel` zwraca dwie dane – obraz etykietowany oraz wartość skalarną – liczbę etykiet, która jest równoznaczna liczbie obiektów na obrazie.

Zadanie 64 *Wyznacz automatycznie liczbę komórek znajdujących się na obrazach* `cells1.bmp` *oraz* `cells2.bmp`*.*

Zadanie 65 *Napisz program, który dla na obrazie* `kostka.bmp` *wyznaczy liczbę ścian kostki wybranego koloru. Działanie programu powinno być następujące. Użytkownik interaktywnie wskazuje myszką obszar (ścianę kostki) o wybranym przez siebie kolorze, program zaś podaje liczbę ścian o tym kolorze znajdujących się na obrazie. Program powinien wykonać segmentację koloru, filtrację obrazu posegmentowanego (w celu usunięcia zbędnych pikseli), oraz etykietowania w celu zliczenia obiektów. Program należy zapisać w formie funkcji o jednym argumencie wejściowym - obrazie, i jednym wyjściowym - liczbie obiektów.*

8.2 Opis obrazu poprzez cechy

Obraz uzyskany w wyniku filtracji, a następnie segmentacji i etykietowania, składa się z obiektów oznaczonych etykietami. Kolejnym etapem w procesie rozpoznawania jest wyznaczenie opisu obrazu. Ściśle rzecz ujmując, wyznaczany opis dotyczy obiektów, uzyskanych w wyniku wykonania wcześniejszych etapów przetwarzania. Opis obiektów powinien umożliwiać ich rozróżnienie bądź ocena jakościowa. Do tego celu wykorzystuje się ich *cechy*. Istnieje wiele różnych rodzajów cech i metod ich wyznaczania. Zbiór cech wyznaczonych dla danego obiektu nazywany jest *wektorem cech*.

W sytuacji, gdy na obrazie znajduje się wiele obiektów uprzednio zaetykietowanych, wektory cech wyznaczane są dla wszystkich obiektów. MATLAB oferuje wygodną instrukcję umożliwiającą otrzymanie macierzy cech, składającej się z wektorów cech wszystkich obiektów obecnych na obrazie. Jest nią instrukcja `regionprops`, która wyznacza wybrane (spośród dostępnych opcji) cechy kolejno dla wszystkich zaetykietowanych obiektów. Instrukcja ta zwraca strukturę, której poszczególne pola odpowiadają wybranym uprzednio cechom. Argumentem tej instrukcji jest obraz z etykietami oraz lista wybranych cech. Na liście tej znajdują się nazwy cech, niektóre spośród nich wymieniono poniżej.

- `Area` Obliczane jest pole powierzchni obiektu. Odpowiednie pole struktury wyjściowej jest wektorem pól powierzchni wszystkich obiektów znajdujących się na obrazie wejściowym.
- `Perimeter` Obliczany jest obwód obiektu. Odpowiednie pole struktury wyjściowej jest wektorem zawierającym długości konturów wewnętrznych wszystkich obiektów znajdujących się na obrazie wejściowym.
- `BoundingBox` Zwracane są parametry najmniejszego prostokąta, w którym w całości mieści się obiekt. Parametrami tymi są kolejno: wsp. x lewego-górnego wierzchołka, wsp. y tegoż wierzchołka, szerokość prostokąta i jego wysokość. Pole struktury wyjściowej jest więc macierzą o liczbie wierszy równej liczbie obiektów i liczbie kolumn równej 4.
- `Centroid` Dla każdego obiektu obliczane są współrzędne jego środka ciężkości: kolejno x i y. Zwracana jest macierz dwukolumnowa.
- `EulerNumber` Wyznaczana jest dla każdego obiektu jego liczba Eulera tj. w tym przypadku jeden minus liczba otworów w obiekcie (np. litera 'L' nie posiada żadnych otworów, zaś litera 'B' posiada dwa otwory - liczba Eulera w pierwszym przypadku wynosi 1, w drugim zaś: -1).
- `Extent` Jest to wartość skalarna wyliczana jako iloraz pola powierzchni obiektu oraz `Area` pola najmniejszego prostokąta `BoundingBox`.
- `EquivDiameter` Zwraca średnicę okręgu o polu równym polu powierzchni obiektu wyznaczaną jako: `sqrt(4*Area/pi)`.

Pełna lista cech jest wyświetlana po wpisaniu komendy `help regionprops` lub `doc regionprops`.

8.3 Prosta klasyfikacja obiektów

Przykład 51 przedstawia proces przetwarzania obrazów komórek (obrazy zostały sztucznie wygenerowane) w celu rozróżnienia dwóch ich rodzajów – normalnych i wydłużonych. Cechą wykorzystaną w przykładzie jest stosunek kwadratu obwodu do pola powierzchni. Cecha taka pozwala na rozróżnienie obiektów o formie wydłużonej od innych o formie bardziej zwartej. Jest ona wyznaczana na podstawie dwóch parametrów odczytywanych przy pomocy komendy `regionprops`: pola powierzchni (`area`) oraz obwodu – długości konturu wewnętrznego (`perimeter`). Przed wyznaczeniem cech następuje wstępne przetwarzanie obrazu składające się z filtracji, segmentacji, kolejnej filtracji obrazu posegmentowanego oraz etykietowania. Po wyznaczeniu cech następuje progowanie wektora zawierającego cechy kolejnych obiektów z ustalonym empirycznie progiem. Wartość progowa została tak dobrana, by obiekty o kształcie wydłużonym (a ściślej rzecz ujmując - ich cechy)

znalazły się po jednej jego stronie, zaś te o kształcie zwartym - po drugiej. Obraz z etykietami jest następnie przetwarzany z wykorzystaniem tablicy korekcji (komenda `intlut`) w obraz zawierający jedynie obiekty o wartości cechy poniżej progu.

Przykład 51 *Wykrywanie komórek wydłużonych i zwartych*

```
>> im = imread('cells1.bmp');
>> imshow(im);title('obraz oryginalny');
>> im2 = imclose(im,ones(3));    % filtracja wstępna
>> im3 = im2 > 85;   % segmentacja przez progowanie
>> im3a = imerode(im3,ones(5)); % otwarcie przez rekonstrukcję
>> im4 = imreconstruct(im3a,im3); % obrazu po segmentacji
>> figure; imshow([im im2 ; im2uint8(im3) im2uint8(im4)]);
>> title('filtracja i segmentacja');
>> [im5 n] = bwlabel(~im4);   % etykietowanie
>> figure; imshow(im5+1,rand(100,3)); title('wynik etykietowania');
>> c = regionprops(im5,'Perimeter','Area');
>> ar = cat(1,c.Area);   % wektor powierzchni obiektów
>> pe = cat(1,c.Perimeter); % wektor obwodów obiektów
>> cc = (pe.*pe)./ar    % wektor współczynników (cech)
>> lut = cc < 14;
>> lut1=zeros(256,1);
>> lut1(1:size(lut,1)) = lut; % tablica korekcji
>> im6 = intlut(uint8(im5-1),uint8(lut1)); % zastosowanie tablicy kor.
>> im7 = (1 - uint8(im4)) + im6;   % przygotowanie do wyświetlenia
>> figure; imshow(im7+1,[0 0 0; 0 0 0 ;1 0 0 ; 0 1 0]); title('wynik klasyfikacji');
```

Pytanie 23 *Jak zapisać ostatnią linię przykładu by komórki były wyświetlane w kolorach zółtym i zielonym ?*

Pytanie 24 *Zwróć uwagę na obraz* `im7` *– co oznaczają poszczególne wartości jego punktów ?*

Zadanie 66 *Napisz funkcję, która wykona zadania z przykładu 51 dla trzech parametrów wejściowych: zmiennej zawierającej obraz początkowy, progu segmentacji (w przykładzie - 85) i progu wektora cech obiektów (w przykładzie - 16). Funkcja powinna zwracać obraz o trzech możliwych wartościach punktów: 0 – dla tła, 1 – dla obiektów o wartości cechy powyżej progu wektora cech obiektów i 2 – dla obiektów o wartości cechy poniżej tego progu.*

Zadanie 67 *Poeksperymentuj wykorzystując funkcję z zadania 66 z różnymi wartościami progowymi.*

Zadanie 68 *Jak należy zmodyfikować funkcję z zadania 66 tak, by było możliwe z jej pomocą znalezienie wydłużonych komórek na obrazie* `cells2.bmp` *? Zapisz osobno zmodyfikowaną funkcję. Dobierz odpowiednie wartości progowe i wykryj oba typy komórek.*

Zadanie 69 *Zapisz funkcje przykładu 51 oraz zadania 68 jako pojedynczą funkcję umożliwiającą segmentację obu obrazów testowych.*

Zadanie 70 *Wykonaj przykład "Identifying Round Objects" dostępny w systemie pomocy do Image Processing Toolbox.*

Zadanie 71 *Wykonaj przykład "Basic Image Enhancement and Analysis Techniques" dostępny w systemie pomocy do Image Processing Toolbox.*

8.4 Identyfikacja większej liczby obiektów na obrazach

Przykłady w poprzednim punkcie ilustrowały przypadek prostej klasyfikacji do jednej z dwóch klas odpowiadających obiektom o cechach spełniających i niespełniających zadany warunek. Warunek ten jest przy tym definiowany poprzez wartość progową wybranej cechy. W przypadku większej liczby klas, niekiedy także wystarczy ustalić pewną liczbę wartości progowych (np. w przykładzie 51 można zdefiniować kilka progów i w ten sposób przypisać komórki do większej liczby kategorii względem ich długości).

W większości jednak przypadków o przynależności obiektów do klas decyduje większa liczba cech. Przykład 52 pokazuje wybrane cechy wyznaczane dla zestawu obiektów – cyfr zapisanych przy pomocy ściśle określonej czcionki (Arial). Cechy poszczególnych cyfr zostaną wyznaczone na podstawie cech uzyskanych przy pomocy komendy `regionprops`. Nie zawsze będą to jednak cechy bezpośrednio uzyskane przy pomocy tej komendy, lecz cechy wyliczone na ich podstawie. Do scharakteryzowania 10 cyfr wykorzystane zostaną następujące cechy:

1. Liczba Eulera – wartość skalarna
2. Extent – wartość skalarna
3. Względne położenie środka ciężkości w najmniejszym prostokącie obejmującym obiekt (`BoundingBox`) – dwuelementowy wektor (położenie względem osi 'x' i 'y').

Dwie pierwsze cechy uzyskuje się bezpośrednio ze struktury wyjściowej komendy `regionprops`. Ostatnia cech jest wyliczana na podstawie cech `Centroid` oraz `BoundingBox`.

Przykład 52 *Wyznaczanie wektorów cech wzorców*

```
>> i = imread('cyfry.tif');
>> [j liczbaklas] = bwlabel(i);
>> imshow(j+1,rand(100,3));
>> cechy = regionprops(j, 'BoundingBox','Centroid','EulerNumber','Extent');
>> euler = cat(1,cechy.EulerNumber)  %  konwersja pola EulerNumber struktury w macierz
>> extent = cat(1,cechy.Extent)   % j.w. pola Extend
>> bb = cat(1,cechy.BoundingBox)  % j.w. pola BoundingBox
>> centroid = cat(1,cechy.Centroid)   % j.w. pola Centroid
>> cposx = (centroid(:,1)-bb(:,1))./bb(:,3) % wzgledna pozycja 'x' sr.ciezkosci
>> cposy = (centroid(:,2)-bb(:,2))./bb(:,4) % wzgledna pozycja 'y' sr.ciezkosci
>> c = [euler extent cposx cposy]    % ostateczne wektory cech znakow
```

Zadanie 72 *Zapisz przykład 52 w postaci funkcji, której argumentem jest zmienna zawierająca obraz binarny, zaś wartością wyjściową – macierz wektorów cech (w przykładzie 52 – macierz* `c`*) .*

Ostatnia macierz wyświetlona w Przykładzie 52 – `c` jest macierzą, której kolumny odpowiadają poszczególnym cechom, zaś wiersze – znakom. Jest to macierz wzorców wartości cech. Będzie ona wykorzystana w procesie rozpoznawania. Dla każdego rozpoznawanego znaku zostanie wyznaczony wektor cech, a następnie zostaną wyznaczone odległości między tym wektorem a wszystkimi wektorami z macierzy `c`. Wektor macierzy cech, dla którego odległość ta jest najmniejsza, odpowiada rozpoznanemu znakowi.

Pytanie 25 *Jak wyznaczana jest względna pozycja środka ciężkości? Jakim wzorem można to opisać? Zapisz ten wzór wykorzystując następujące oznaczenia: x_C, y_C współrzędne środka ciężkości, x_P, y_P położenie lewego-górnego wierzchołka najmniejszego prostokąta, dx, dy - rozmiary tegoż.*

Pytanie 26 *Czy cechy wchodzące w skład wektora cech wykorzystywanego w przykładach są niezmiennicze względem 1) przesunięcia 2) zmiany skali 3) obrotu ? Odpowiedź uzasadnij. Przez niezmienniczość względem operacji rozumiemy fakt, iż po wykonaniu danej operacji na rozpoznawanym obiekcie wartości cech nie ulega zmianie.*

Pytanie 27 *Jakie mógłbyś zaproponować inne rodzaje cech?*

Zadanie 73 *Wyznacz macierz* `c` *z Przykładu 52 dla innych cech wyznaczanych przy pomocy komendy* `regionprops`*. Skorzystaj z cech wymienionych powyżej lub innych, opisanych w pomocy dla tej komendy. Zwróć uwagę na to czy wektory cech pozwalają na jednoznaczne rozróżnienie poszczególnych znaków.*

Po przetworzeniu obrazu zgodnie z opisanymi powyżej etapami, uzyskuje się obraz składający się z wyraźnie wyodrębnionych i – jeśli istnieje taka potrzeba – opisanych etykietami obiektów. Ponadto znany jest też opis tych obiektów przy pomocy wektora cech. Na podstawie tych danych podejmowana jest decyzja – rozpoznawany obiekt jest przypisywany tej klasie, której cechy wzorcowe są najbliższe cechom rozpoznawanego obiektu. Do określenia przynależności obiektu do klasy, w tego typu rozpoznawaniu stosowane są metody klasyfikacji.

Przykład 53 jest kontynuacją Przykładu 52. W poprzednim przykładzie został wyznaczony wektor cech wzorców cyfr. W tym pokazany zostanie proces rozpoznawania poprzedzony filtracją, segmentacją i wyznaczeniem cech obrazu rozpoznawanego. Obraz wejściowy w tym przypadku nie jest binarnym obrazem zawierającym wyraźne i czytelne litery. Tym razem jest to zaszumiony obraz w skali szarości. W związku z tym wymaga zastosowania filtracji przez segmentacją. Przefiltrowany i posegmentowany obraz jest etykietowany. Następnie zostaje wyznaczona macierz zawierająca wektory cech poszczególnych znaków (w sposób analogiczny do pokazanego w Przykładzie 52). Dzięki temu znany jest opis obrazu. Opis ten jest następnie wykorzystywany przez klasyfikator najbliższego sąsiada.

Przykład 53 *Rozpoznawanie znaków*

```
>> % przykład wymaga wcześniejszego wyznaczenia macierzy cech
>> %  z poprzedniego przykładu
>> o = imread('cyfry2.tif');
>> imshow(o);   % zaszumiony obraz do rozpoznania
>> o1 = imclose(o,ones(3)); % filtracja morfologiczna
>> t = graythresh(o1)  % wyznaczenie progu binaryzacji
>> o2 = im2bw(o1,t);   % segmentacja przez progowanie
>> o2 = imcomplement(o2);  % odwróć obraz
>>  % wyznaczanie opisu obrazu (wektorów cech)
>> [o3 lob] = bwlabel(o2);  % etykietuj,  lob - liczba obiektów
>> imshow(o3+1,rand(100,3));  % wyswietl w losowych barwach
>> cechy = regionprops(o3, 'BoundingBox','Centroid','EulerNumber','Extent');
>> euler = cat(1,cechy.EulerNumber)  % liczba Eulera
>> extent = cat(1,cechy.Extent)   %
>> bb = cat(1,cechy.BoundingBox)  % parametry najmniejszych prostokatow
>> centroid = cat(1,cechy.Centroid)   % wspolrzedne sr.ciezkosci
>> cposx = (centroid(:,1)-bb(:,1))./bb(:,3) % wzgledna pozycja 'x' sr.ciezkosci
>> cposy = (centroid(:,2)-bb(:,2))./bb(:,4) % wzgledna pozycja 'y' sr.ciezkosci
>> cechy_znakow = [euler extent cposx cposy]    % ostateczne wektory cech znakow
>> % rozpoznawanie
>> for k=1:1:lob   % dla wszystkich znaków znalezionych na obrazie wykonuj
>>  aktualny = cechy_znakow(k,:);  % wekt. cech aktualnie rozpozn.znaku
>>  d = sqrt(sum(abs(c - repmat(aktualny,liczbaklas,1)).^2,2)); % wektor odleglosci
>>  znak = find (d == min(d));  % znajdź najmniejszą odległość
```

```
>>  znak - 1   % wydrukuj rozpoznana cyfre
>> end
```

Pytanie 28 *Jaką inną miarę odległości można zastosować w procesie rozpoznawania zamiast odległości Euklidesowej? Jak wyglądałby odpowiedni fragment kodu MATLAB-a?*

Operacja wyznaczania wektorów cech jest wykonywana dwukrotnie: na etapie uczenia (czyli zapamiętywania wektorów wzorcowych) oraz właściwego rozpoznawania (gdy wyznaczamy wektory cech rozpoznawanych znaków). Warto więc zastosować funkcję z zadania 72.

Zadanie 74 *Zmodyfikuj kod przykładu 53, tak by wykorzystać funkcję z zadania 72.*

W procesie rozpoznawanie może zdarzyć się, iż pomimo skutecznej filtracji, po segmentacji na obrazie znajdują się pojedyncze punkty lub niewielkie zgrupowanie punktów, nie będące rozpoznawanymi obiektami. W celu ich usunięcia możliwe dokonuje się modyfikacji sposobu filtracji przed segmentacją lub wykonuje dodatkową filtrację po segmentacji. Innym rozwiązanie wymaga zmiany algorytmu na etapie rozpoznawania. Można wówczas wprowadzić dodatkowy parametr definiowany przez użytkownika – próg wielkości obiektu. Dzięki jego wprowadzeniu, w dodatkowej instrukcji warunkowej w głównej pętli rozpoznawania rozpoznawane są jedynie te obiekty, których pole powierzchni jest większe od zadanego progu. Zmodyfikowaną pętlę rozpoznawania pokazuje Przykład 54.

Przykład 54 *Rozpoznawanie z progiem wielkosci obiektu*

```
>> % rozpoznawanie
>> cechy1 = regionprops(o3,'Area'); % pobierz dodatkowe cechy - pola powierzchni
>> prog = 15  % wartosc progowa pola powierzchni
>> pole = cat(1,cechy1.Area);   % wektor pol powierzchni
>> for k=1:1:lob   % dla wszystkich obiektow znalezionych na obrazie wykonuj
>>  if (pole(k,1) > prog)  % prog wielkosci obiektu
>>   aktualny = cechy_znakow(k,:);  % wekt. cech aktualnie rozpozn.znaku
>>   d = sqrt(sum(abs(c - repmat(aktualny,liczbaklas,1)).^2,2)); % wektor odleglosci
>>   znak = find (d == min(d));  % znajdź najmniejszą odległość
>>   znak - 1   % wydrukuj rozpoznana cyfre
>>  else
>>   disp('obiekt pominiety');
>>  end
>> end
```

Zadanie 75 *Zaprojektuj etap filtracji i segmentacji dla obrazów* `cyfry1.tif`, `cyfry3.tif` *oraz* `cyfry4.tif` *tak by cyfry znajdujące się na tych obrazach były poprawnie rozpoznawane.*

9 Przekształcenia geometryczne

9.1 Rozmiar obrazu i jego zmiana

Rozmiar obrazu zależy od parametrów sensora wizyjnego z którego obraz ten pochodzi. Rozmiar ten może być dowolnie modyfikowany. Każde zwiększenie rozmiaru wymaga jednak zastosowania interpolacji dzięki której mogą być uzupełnione brakujące wartości pikseli na obrazie wynikowym. Najprostszymi metodami interpolacji są metody: najbliższego sąsiada, dwuliniowa i dwukwadratowa. W przypadku metody najbliższego sąsiada brakująca wartość piksela jest równa wartości najbliższego spośród sąsiadów o znanych wartościach. W metodzie dwuliniowej przeprowadzana

jest interpolacja liniowa na podstawie wartości czterech najbliższych sąsiadów. W metodzie dwukwadratowej funkcja interpolująca jest funkcja kwadratową, wykorzystującą większą liczbę najbliższych sąsiadów. Do zmiany rozmiaru (przeskalowania) obrazu przeznaczona jest w MATLAB-ie komenda `imresize`. Jej argumentami są obraz przeznaczony do przeskalowania oraz parametry skalowania, którym mogą być m.in.:

- współczynnik skalujący (pojedyncza wartość), lub
- rozmiary obrazu po przeskalowaniu (wektor), i dodatkowo:
- metoda przeskalowania: (m.in. `'nearest'`,`'bilinear'`,`'bicubic'`)

Przykład 55 *Zmiana rozmiaru macierzy obrazu*

```
im = checkerboard(10);  % sztucznie utworzona macierz obrazu
im1 = imresize(im,4,'nearest');   % metoda najbliższego sąsiada
im2 = imresize(im,4,'bilinear');  % metoda dwuliniowa
im3 = imresize(im,4,'bicubic');   % metoda dwukwadratowa
imshow([im1 im2 im3])
```

Pytanie 29 *Porównaj obrazy* `im` *oraz* `im1` *i* `im2` *- czym różnią się oba rodzaje powiększania obrazu ? Która metoda daje – Twoim zdaniem – najlepsze wyniki ?*

Zadanie 76 *Wykonaj identyczne przeskalowanie obrazów ale dla obrazu obróconego o 30 i 45 stopni. Jak oceniłbyś teraz zastosowane meotdy ?*

Zadanie 77 *Wykonaj eksperymenty z przeskalowaniem z różnymi współczynnikami dla dowolnego, wczytanego z dysku obrazu, porównaj wyniki dla trzech metod skalowania.*

9.2 Podstawowe przekształcenia geometryczne

Przekształcenia geometryczne zmieniają położenie punktów obrazu. Przekształcenia geometryczne obrazu mogą być wykonywane na dwa sposoby. Pierwszy polega na wyznaczaniu położenia każdego piksela obrazu wejściowego na obrazie wyjściowym. Drugi zaś dla każdego piksela obrazu wyjściowego znajduje jego odpowiednik na obrazie wejściowym. Pierwszy rodzaj przekształcenia nazywany jest przekształceniem prostym, drugi natomiast - odwrotnym.

Do przekształceń geometrycznych obrazu zaliczamy: przesunięcie, obrót, przeskalowanie i pochylenie. Przekształcenia te są szczególnymi przypadkami przekształcenia afinicznego, które można zapisać jako (przekształcenie proste):

$$[x, y, 1] = [u, v, 1] \cdot \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix}, \tag{22}$$

przy czym u, v są współrzędnymi punktu obrazy wejściowego, x, y – odpowiadającemu mu punktu obrazu wyjściowego, zaś występująca we wzorze macierz jest macierzą przekształcenia afinicznego o sześciu parametrach: $a_{11}, a_{12}, a_{21}, a_{22}, a_{31}, a_{32}$, od których wartości zależy typ przekształcenia. Wartości współczynników dla poszczególnych rodzajów przekształceń są umieszczone w tablicy 1.

Przekształcenie odwrotne jest opisane zależnością współrzędnych punktu obrazu wejściowego od współrzędnych odpowiadającemu mu punktu obrazu wyjściowego:

$$[u, v, 1] = [x, y, 1] \cdot \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix}^{-1}. \tag{23}$$

nazwa	a_{11}	a_{12}	a_{21}	a_{22}	a_{31}	a_{32}	**uwagi**
identyczność	1	0	0	1	0	0	bez zmian obrazu
przesunięcie	1	0	0	1	t_x	t_y	o wektor $[t_x, t_y]$
skalowanie	s_x	0	0	s_y	0	0	z współczynnikami $[s_x, s_y]$
obrót	$\cos\theta$	$\sin\theta$	$-\sin\theta$	$\cos\theta$	0	0	o kąt θ
pochylenie poziome	1	0	a	1	0	0	ze współczynnikiem a
pochylenie pionowe	1	a	0	1	0	0	ze współczynnikiem a

Tabela 1: Współczynniki różnych przekształceń afinicznych (równ. 22).

Pytanie 30 *Przyjmując oznaczenia wektora przesunięcia, kąta obrotu oraz współczynników skalujących takie jak w Tablicy 1, zapisz kolejno parametry tych operacji w wersji dla przekształcenia odwrotnego.*

Przekształcenia geometryczne obrazu w MATLAB-ie wykorzystują specjalną strukturę danych `tform`, w której zapisywany jest typ przekształcenia oraz jego parametry. Jednym z przekształceń, które moga być wykonywane z wykorzystaniem tej struktury jest przekształcenie afiniczne. Funkcją, która tworzy tego typu strukturę na podstawie macierzy przekształcenia afinicznego (równ. 22) jest `maketform`. Z kolei zastosowanie odpowiedniego przekształcenia na obrazie wymaga użycia funkcji `tformfwd` dla przekształcenia prostego oraz `tforminv` – dla odwrotnego.

Przykład 56 *Przekształcenie afiniczne – przesunięcie*

```
t = [1 0 0; 0 1 0; 15 30 1]
tf = maketform('affine',t)
im = checkerboard(30)
im2 = imtransform(im,tf,'XData',[1 size(im,2)],'Ydata',[1 size(im,1)],'FillValue',0.5);
imshow(im2)
```

Parametry wywołania komendy `imtransform 'XData'` oraz `'YData'` określają obszar na jakim jest wykonywane przekształcenie afiniczne. Te dwa parametry należy wykorzystywać jedynie dla przesunięcia. Parametr 'FillValue' o wartości 0.5 określa odcień szarości, który pojawi się na obszarze odsłoniętym po przesunięciu obrazu.

Zadanie 78 *Wykonaj pozostałe przekształcenia afiniczne dla obrazu szachownicy oraz dla dowolnego obrazu rzeczywistego.*

Przekształcenie polegające na pojedynczym obrocie.

Przekształcenia afiniczne podstawowe mogą być dowolnie łączone. Macierz przekształcenia złożonego jest równa iloczynowi macierzy pojedynczych przekształceń:

$$A = A_1 \cdot A_2 \cdot \ldots \cdot A_n, \tag{24}$$

przy czym A jest macierzą przekształcenia złożonego, a $A_1 \ldots A_n$ są macierzami przekształceń pojedynczych.

Przykład 57 *Przekształcenie afiniczne – obrót*

```
kat = pi/6
r = [cos(kat) sin(kat) 0; -sin(kat) cos(kat) 0; 0 0 1]
s = [0.5 0 0; 0 1.5 0; 0 0 1]
a = r*s
tf = maketform('affine',a);
im = checkerboard(30);
im2 = imtransform(im,tf,'FillValue',0.5);
imshow(im2);
```

Zadanie 79 *Jakie przekształcenia afiniczne zostały wykonane w Przykładzie 56 ? Wykonaj tę samą operację stosujac przekształcenia kolejno.*

Zadanie 80 *Jak zapisać przekształcenie polegające na wykonaniu obrotu o zadany kąt* `k` *wokół zadanego punktu* `[x y]` *? Wykonaj stosowne eksperymenty.*

Zadanie 81 *Wykonaj przykład "Creating a Gallery of Transformed Images" dostępny w systemie pomocy do Image Processing Toolbox.*