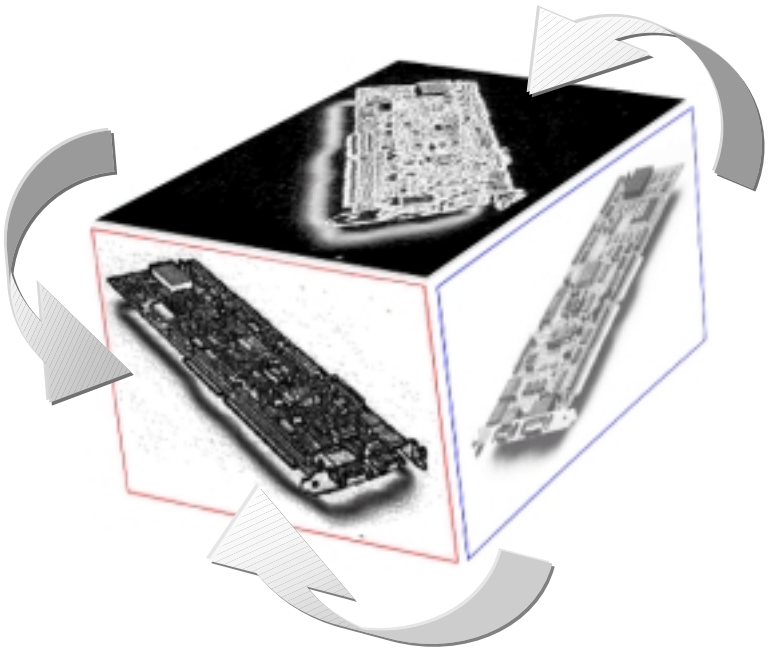
Politechnika Łódzka
Instytut Elektroniki

Laboratorium przetwarzania obrazów



Autorzy opracowania:

P. Pełczyński, P. Strumiłło, M. Strzelecki

Łódź, październik 2000

Spis treści:

Biblioteka Przetwarzania Obrazów (*Image Processing Toolbox - IPT*) pakietu matematycznego MATLAB®

Cel ćwiczenia

Zapoznanie z biblioteką przetwarzania obrazów (*Image Processing Toolbox - IPT*) pakietu matematycznego MATLAB.

Wiadomości wstępne

Image Processing Toolbox (IPT) jest biblioteką pakietu MATLAB zawierającą zbiór specjalizowanych funkcji przeznaczonych do przetwarzania i analizy obrazów. Funkcje tej biblioteki umożliwiają, m.in., wykonywanie następujących działań na obrazach:

- przekształcenia geometryczne obrazów,
- projektowanie filtrów i przestrzenna filtracja liniowa obrazów,
- transformacje obrazów (Fouriera, kosinusowa),
- poprawa jakości obrazu,
- analiza obrazu,
- przetwarzanie obrazów binarnych.

W celu sprawdzenia, czy używana przez Ciebie instalacja pakietu MATLAB jest wyposażona w bibliotekę IPT, w oknie komend (tj. MATLAB Command Window), wpisz polecenie:

```
ver
```

Po zastosowaniu komendy `ver` zostanie wyświetlona informacja o numerze wersji oraz lista bibliotek, w którą jest wyposażona używana przez Ciebie licencja pakietu MATLAB. W wyświetlonej liście powinna znajdować się m.in. pozycja:

```
Image Processing Toolbox Version 2.1    15-Dec-1997
```

Większość funkcji biblioteki IPT to m-pliki napisane w języku skryptowym pakietu MATLAB (tj. pliki z rozszerzeniem `*.m`). Działanie poszczególnych funkcji i ich składnie można uzyskać za pomocą polecenia

```
help function_name
```

np. podając polecenie `help imread` uzyskasz informację o funkcji `imread`, służącej do ładowania obrazów z plików dyskowych do pamięci roboczej pakietu MATLAB. Zapoznaj się ze składnią tej funkcji i typami plików graficznych, które mogą być ładowane do pamięci roboczej pakietu.
Pełniejszy opis poszczególnych poleceń można również uzyskać korzystając z pomocy interaktywnej dostępnej w menu głównym okna komend programu MATLAB (np. `Help>Help Desk HTML`).

Kody źródłowe funkcji zapisanych w m-plikach można wyświetlić stosując polecenie:

```
type function_name
```

np. wprowadź polecenie `type filter2`, by zapoznać się z kodem źródłowym funkcji służącej do dwuwymiarowej filtracji obrazu.

Pełną listę funkcji dostępnych w bibliotece IPT wyświetla polecenie:

```
help images/contents
```

Z działaniem niektórych funkcji i możliwościami biblioteki IPT pakietu MATLAB możesz się zapoznać dokładniej podając polecenie:

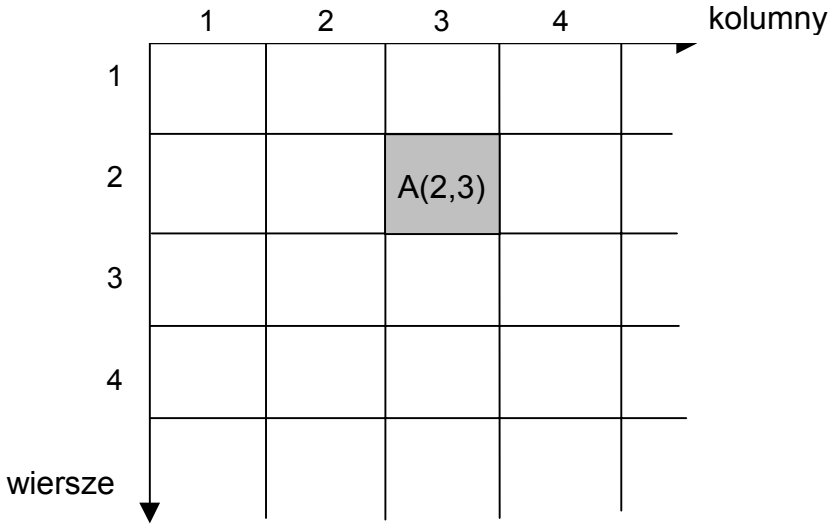
```
demo
```

oraz wybierając pozycję Image Processing Toolbox z menu Toolboxes.

Okno demonstracyjne pakietu MATLAB (MATLAB Demo Window) zawiera również przykłady działania innych bibliotek oraz wprowadzenie do podstawowych funkcji przetwarzania numerycznego i funkcji graficznych dostępnych w pakiecie MATLAB.

Struktury danych stosowane do reprezentacji i przetwarzania obrazów w bibliotece *Image Processing Toolbox*

Podstawowe typy obrazów monochromatycznych są reprezentowane w pakiecie MATLAB za pomocą tablic dwuwymiarowych. Każdy element tablicy odpowiada jednemu punktowi obrazu cyfrowego określanego jako *piksel*. Wartości elementów obrazu reprezentują jasność poszczególnych pikseli obrazu, np. jasność punktu obrazu znajdującego się na przecięciu drugiego wiersza i trzeciej kolumny tablicy A można uzyskać odczytując wartość elementu tablicy A o współrzędnych (2,3), tj. A(2,3), gdzie 2 i 3 są indeksami wskazującymi na ten punkt obrazu patrz rys. 1.



Rys. 1. Obraz jako tablica dwuwymiarowa

W pakiecie MATLAB, elementy tablic są standardowo reprezentowane za pomocą 64 bitowych liczb zmiennoprzecinkowych (klasa `double`). Zastosowanie takiej precyzji do kodowania obrazów nie jest konieczne i zajmuje bardzo duży obszar pamięci operacyjnej komputera. Z powyższych względów, do kodowania obrazów w bibliotece IPT stosuje się głównie klasę `uint8`, tj. zmienną ośmiobitową bez znaku (zauważmy, że do wyświetlania obrazów nie ma potrzeby stosowania zmiennych przyjmujących ujemne wartości).

Zmienne klasy `uint8`

Biblioteka IPT udostępnia ograniczony zbiór działań na zmiennych klasy `uint8`, tj.:

- wyświetlanie obrazów reprezentowanych za pomocą zmiennej `uint8`,
- indeksowanie (adresowanie punktów) obrazów,
- zmianę wymiarów tablic i kolejności elementów tablic, m.in. za pomocą poleceń: `reshape, cat, permute`.

Niezależnie od klasy zmiennej stosowanej do kodowania jasności punktu obrazu, w bibliotece IPT stosuje się cztery podstawowe struktury danych do reprezentacji obrazów:

- obrazy indeksowane (*indexed images*),
- obrazy monochromatyczne (*intensity images*),
- obrazy binarne (*binary images*),
- obrazy RGB (*RGB images*).

Obrazy indeksowane (*indexed images*)

Obrazy indeksowane składają się z dwóch typów tablic:

- trójkolumnowej tablicy (mapy) kolorów,
- dwuwymiarowej tablicy (typu `double` lub `uint8`) indeksów do tablicy kolorów o rozmiarze odpowiadającym rozmiarowi obrazu.

Tablica kolorów jest trójkolumnową tablicą typu `double`. Każdy wiersz tej tablicy zawiera wartość odpowiednio czerwonej (**R**ed), zielonej (**G**reen) i niebieskiej (**B**lue) składowej koloru, z których każda może przyjmować wartości z zakresu [0, 1]. Maksymalna liczba wierszy tablicy kolorów wynosi 256.

Tablica indeksów zawiera indeksy wskazujące na wiersze tablicy kolorów, przy czym adresowanie tablicy kolorów zależy od tego czy tablica indeksów jest klasy `double` czy `uint8`. Jeżeli tablica indeksów `A` jest klasy `double` i np. `A(2,3)=10` to kolor tego punktu obrazu jest wyznaczony przez zawartość dziesiątego wiersza tablicy kolorów. Jeżeli zaś tablica indeksów jest klasy `uint8` to kolor punktu obrazu `A(2,3)` jest wyznaczony przez zawartość jedenastego wiersza tablicy kolorów (indeksowanie rozpoczyna się od indeksu 0, tj. indeks 0 wskazuje na pierwszy wiersz tablicy kolorów).

Klasę zmiennej przechowywanej w obszarze roboczym pakietu MATLAB, możesz sprawdzić np. za pomocą polecenia `whos`, które wyświetla nazwę, rozmiar i klasę aktualnie używanych zmiennych utworzonych w obszarze roboczym (ang. *MATLAB workspace*).

Obrazy monochromatyczne (*intensity images*)

Obrazy monochromatyczne są reprezentowane w bibliotece IPT za pomocą pojedynczej tablicy, której elementy odpowiadają jasności poszczególnych punktów obrazu. Tablica obrazu monochromatyczny może być klasy `double` lub `uint8`. Dla klasy `double` elementy tablicy przyjmują wartości z zakresu [0, 1], zaś dla tablicy klasy `uint8` wartości całkowite z zakresu [0, 255]. Wartość 0 odpowiada punktowi o zerowej jasności (tj. punktowi czarnemu) a wartość 1 (lub 255) odpowiada jasności maksymalnej.

Obrazy binarne *(binary images)*

Punkty obrazu binarnego przyjmują jedną z dwóch dyskretnych wartości. Podobnie jak dla obrazów monochromatycznych tablice reprezentujące obrazy binarne są klasy `double` lub `uint8`. Zaleca się stosowanie głównie klasy `uint8` ze względu na oszczędność pamięci. Funkcje biblioteki IPT, które w wyniku zwracają obrazy binarne stosują klasę `uint8`.

Obrazy RGB *(RGB images)*

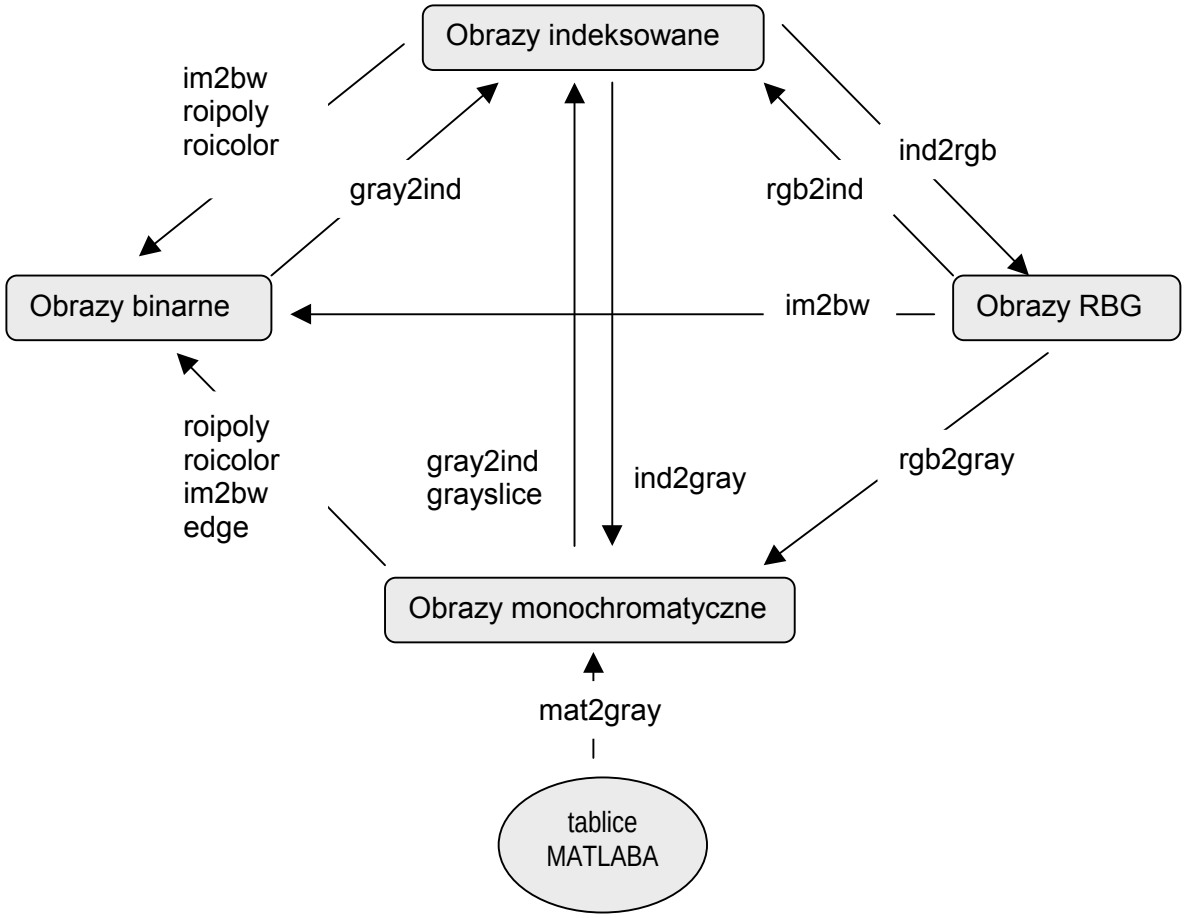
Obraz RBG jest definiowany przez trzy oddzielne tablice, każda o wymiarach odpowiadających wymiarowi obrazu. Tablice te zawierają intensywności kolorów składowych kolejno: czerwonego (**R**ed), zielonego (**G**reen) i niebieskiego (**B**lue). Zatem obraz RGB jest tablicą trójwymiarową $M \times N \times 3$, gdzie M jest liczbą wierszy a N liczbą kolumn obrazu, a trzeci wymiar tablicy wskazuje na składową koloru. Zatem kolor każdego punktu obrazu jest wynikiem złożenia trzech kolorów składowych. Tablica obrazu RGB jest klasy `double` lub `uint8`. Dla tablicy klasy `double` elementy tablicy przyjmują wartości z zakresu [0, 1], zaś dla tablicy klasy `uint8` wartości całkowite z zakresu [0, 255].

Biblioteka IPT umożliwia konwersję struktur danych stosowanych do reprezentacji obrazów. Rys. 2 ilustruje zbiór funkcji służących do tego celu.

W pakiecie MATLAB dla klasy zmiennych `uint8` jest dostępny tylko ograniczony zbiór działań wymieniony poniżej:

- wyświetlanie obrazów,
- indeksowanie tablic,
- funkcje `all` i `any`,
- zmiana wymiarów tablic, łączenie tablic itp., (funkcje `reshape`, `cat`, `permute`),
- zapisywanie i odczyt plików binarnych `*.mat`,
- operacje logiczne.

Pamiętaj, że pakiet MATLAB, nie zezwala na wykonywanie innych działań na zmiennych klasy `uint8` niż te, które wymieniono wyżej, np. niedozwolone jest wykonywanie działań arytmetycznych. Rozszerzenie listy dostępnych funkcji, np. arytmetycznych, wymaga wykonania przekształcenia klasy `uint8` do klasy `double`. W tabeli 1 podano przykłady poleceń przekształcenia klas zmiennych dla trzech typów obrazów, stosowanych w bibliotece IPT:



Rys. 2. Funkcje biblioteki IPT do konwersji struktur danych obrazowych

Tabela 1. Funkcje do konwersji klas zmiennych

Typ obrazu	**uint8 → double**	**double → uint8**
indeksowany	`B=double(A)+1;`	`B=uint8(round(A-1));`
monochromatyczny lub RGB	`B=double(A)/255;`	`B=uint8(round(A*255));`
binarny	`B=double(A);`	`B=logical(uint8(round(A-1)));`

Pamiętaj również że:

- dla obrazów indeksowanych konwersja dotyczy tylko tablicy indeksów a nie tablicy kolorów,
- w pakiecie MATLAB mapy kolorów są zawsze klasy `double`,
- obraz indeksowany nie może być poddany konwersji do klasy `uint8` jeżeli elementy tablicy zawierają wartości większe od 256.

Załaduj obraz do obszaru roboczego programu oraz dokonaj konwersji obrazu indeksowanego do obrazu monochromatycznego stosując sekwencje poleceń:

```
[x,map]=imread('trees.tif'); %czytaj plik obrazowy 'trees.tif'
I=ind2gray(x,map);        % konwertuj na obraz monochromatyczny
imshow(x,map);            % wyświetl obraz indeksowany X
figure,imshow(I);         % wyświetl obraz monochromatyczny I
```

Dokonaj konwersji odwrotnej korzystając z polecenia `gray2ind` i porównaj uzyskane wyniki.

Zapoznaj się z funkcją `im2bw` oraz dokonaj binaryzacji obrazu indeksowanego zgodnie z pokazaną procedurą. Następnie zastosuj inną wartość progu i oceń czytelność uzyskiwanych obrazów binarnych

```
[x,map]=imread('trees.tif');   % czytaj plik obrazowy
                               % 'trees.tif'
BW = im2bw(x,map,0.4);         % konwertuj na obraz binarny
imshow(x,map);                 % wyświetl obraz źródłowy
figure, imshow(BW);            % wyświetl obraz binarny
```

Czytanie i zapisywanie plików obrazowych oraz wyświetlanie obrazów za pomocą funkcji biblioteki *Image Processing Toolbox*

Do czytania plików obrazowych (tj. ładowania danych obrazowych do obszaru roboczego pakietu MATLAB) służy funkcja `imread`. W tabeli 2 zebrano typy plików graficznych obsługiwanych przez funkcję `imread`.

Tabela 2. Formaty plików graficznych czytane przez pakiet IPT MATLAB

Format	**Typ pliku**
`'bmp'`	Windows Bitmap (BMP)
`'hdf'`	Hierarchical Data Format (HDF)
`'jpg'` lub `'jpeg'`	Joint Photographic Experts Group (JPEG)
`'pcx'`	Windows Paintbrush (PCX)
`'tif'` lub `'tiff'`	Tagged Image File Format (TIFF)
`'xwd'`	X Windows Dump (XWD)

Sprawdź składnię funkcji `imread` i załaduj do pamięci roboczej pakietu MATLAB obraz `'flowers.tif'`. Po sprawdzeniu klasy zmiennej reprezentującej obraz (np. poleceniem `whos`), dokonaj jego konwersji obrazu na typ monochromatyczny a potem na typ indeksowany. Przy konwersji do obrazu indeksowanego zmniejszaj liczbę wierszy tablicy kolorów i zanotuj liczbę wierszy dla której zauważasz pogorszenie jakości otrzymywanego obrazu.

Do zapisywania obrazów w plikach dyskowych służy funkcja `imwrite`. Funkcja ta może zapisywać formaty graficzne plików takie same jakie może czytać funkcja `imread`.

Funkcja `imshow` biblioteki IPT służy do wyświetlania obrazów na ekranie monitora. Jej składnia zależy od typu wyświetlanego obrazu.

Wyświetlanie obrazów indeksowanych

W celu wyświetlenia obrazu indeksowanego `X` należy zastosować funkcję `imshow` z dwoma parametrami specyfikującymi odpowiednio tablicę indeksów oraz mapę kolorów:

```
imshow(X,map)
```

Należy pamiętać, że zależnie od klasy tablicy indeksów (`uint8` lub `double`) stosowany jest inny schemat indeksowania tablicy kolorów.

Wczytaj i wyświetl zawartość pliku `'trees.tif'` oraz sprawdź klasę tablicy indeksów. Dokonaj konwersji klasy z `uint8` na `double` stosując polecenie:

```
X=double(X)
```

Ponownie wyświetl zawartość tablicy i zwróć uwagę na zafałszowanie kolorów wyświetlanego obrazu. Dokonaj prawidłowej konwersji klasy zmiennej stosując odpowiednie polecenie z tabeli 1. Wyświetl i oceń otrzymany wynik.

Wyświetlanie obrazów monochromatycznych

Obraz monochromatyczny `I`, znajdujący się w obszarze roboczym programu, można wyświetlać stosując polecenie:

```
imshow(I)
```

Dla 24-bitowej grafiki kolorowej, liczba poziomów jasności dla obrazów monochromatycznych wynosi 256. Dla innych ustawień kodowania kolorów liczba dostępnych poziomów szarości wynosi 64.
Polecenie `imshow` umożliwia również wyświetlanie obrazów monochromatycznych z liczbą poziomów szarości zadaną przez użytkownika, np.

```
imshow(I,16)
```

Napisz procedurę wyświetlającą obraz monochromatyczny `'cameraman.tif'` z kolejno zmniejszaną liczbą poziomów szarości (tj., od 50 do 30). Zapamiętaj przy jakiej rozdzielczości zauważasz subiektywne pogorszenie jakości obrazu.

Wyświetlanie obrazów binarnych

Obraz binarny `BW`, znajdujący się w obszarze roboczym programu, wyświetla polecenie:

```
imshow(BW)
```

W IPT wszystkie komendy zwracające w wyniku obraz binarny stosują typ `uint8`. Załaduj jeden z obrazów dostępnych w pakiecie IPT, dokonaj jego konwersji na typ binarny i wyświetl go. Na obrazach binarnych możesz wykonywać działania logiczne, np. wyświetl negatyw obrazu binarnego stosując polecenie `imshow(~BW)`.

Wyświetlanie obrazów RGB

Obraz typu RGB `KOLOR`, znajdujący się w obszarze roboczym programu, wyświetla polecenie:

```
imshow(KOLOR)
```

Dla obrazu RGB klasy `double` elementy trójwymiarowej tablicy obrazu przyjmują wartości z zakresu [0, 1], zaś dla klasy `uint8` wartości z zakresu liczb całkowitych [0, 255].

Wczytaj do pamięci roboczej pakietu obraz typu RGB (np. `flowers.tif`), sprawdź klasę w jakiej reprezentowane są elementy tablicy obrazu. Wykonaj odpowiednią konwersje klasy na inną (tabela 1) tak by komenda `imshow` prawidłowo wyświetlała obrazy niezależnie od klasy zastosowanej zmiennej.

Dodatkowe funkcje wyświetlania i formatowanie obrazów

Funkcja `imshow` umożliwia również bezpośrednie wyświetlenie obrazów zapamiętanych w plikach dyskowych pod warunkiem, że należą do formatów plików graficznych obsługiwanych przez IPT. Dla przykładu polecenie:

```
imshow flowers.tif
```

wyświetli obraz bezpośrednio z pliku `flowers.tif` z pominięciem ładowania tego obrazu do obszaru roboczego pakietu.

Instrukcja `imfinfo` umożliwia uzyskanie wyczerpujących informacji o obrazie zapamiętanym w danym formacie pliku graficznego (uzyskaj te informacje podając polecenie):

```
imfinfo('flowers','tif')
```

Należy podkreślić, że korzystając z funkcji pakietu IPT można wyświetlać również dowolne tablice dwuwymiarowe. Możesz przekonać się o tym wpisując polecenia:

```
x=rand(300,300);
imshow(x)
```

z których pierwsze generuje tablicę o wymiarze 300×300 składającą się z elementów o wartościach losowych a drugie polecenie wyświetla tablicę w postaci obrazu (możesz stosować ten sposób do symulowania zakłóceń szumowych w przetwarzanych obrazach). Wykonaj samodzielnie taką symulację stosując funkcję `randn` do generowania losowego rozkładu Gaussowskiego (pamiętaj, że wyświetlane próbki rozkładu powinny być równe lub większe od zera). Możesz tu skorzystać z dodatkowych parametrów akceptowanych przez funkcję `ismhow` odpowiednio skalujących mapę kolorów. Składnia funkcji `ismhow` z dodatkowymi parametrami jest postaci:

```
imshow(x, [low high])
```

gdzie `low` i `high` definiują zakres wartości skalowany do zakresu [0, 1] dla klasy `double` i skalowanych do zakresu [0, 255] dla klasy `uint8`. Dodatkowo funkcja postaci

```
imshow(x, [])
```

automatycznie skaluje wartości wyświetlanej struktury danych do odpowiedniego zakresu.

Po wykonaniu powyższych ćwiczeń zapoznaj się z funkcją `imnoise` przeznaczoną do generowania różnych zakłóceń addytywnych w obrazach.

Często się zdarza, że samodzielnie opracowujesz procedurę przetwarzania obrazu, a potem odkryjesz, że jest ona dostępna w zbiorze funkcji pakietu IPT. Pamiętaj zatem skrupulatnie czytać `help` i wykorzystywać funkcje przetwarzania danych (np. obrazowych), w które bogato jest wyposażony pakiet IPT.

Sesję zapoznawania się z możliwościami pakietu Image Processing Toolbox zakończ praktycznymi ćwiczeniami z następującymi funkcjami:

- `imresize` % powiększanie/zmniejszanie obrazu
- `zoom` % interaktywne powiększanie/zmniejszanie obrazu
- `imrotate` % obrót wyświetlanego obrazu o zadany kąt
- `imcrop` % interaktywny wybór prostokątnego obszaru analizy obrazu

Reprezentacje obrazów kolorowych

Cel ćwiczenia

Celem ćwiczenia jest poznanie sposobów reprezentowania obrazów barwnych w postaci cyfrowej w komputerze oraz operacji przetwarzania tego typu obrazów i konwersji pomiędzy różnymi systemami ich reprezentacji. Ćwiczenie bazuje na wykorzystaniu funkcji biblioteki *Image Processing Toolbox (IPT)* pakietu MATLAB. Przed przystąpieniem do ćwiczenia należy zapoznać się z teoretycznymi podstawami tworzenia i widzenia obrazów barwnych.

Systemy reprezentacji obrazów barwnych

Obrazy kolorowe są reprezentowane w środowisku MATLAB na dwa różne sposoby:

- Obrazy RGB o pełnej skali kolorów
- Obrazy indeksowane

Reprezentacja RGB (ang. *Red, Green, Blue*) nawiązuje do naturalnego sposobu postrzegania barw przez człowieka. Każdy punkt obrazu jest reprezentowany za pomocą trzech wartości oznaczających intensywności kolorów: czerwonego, zielonego i niebieskiego. Obrazy indeksowane składają się z mapy kolorów i tablicy danych obrazowych, będących indeksami do powyższej mapy. Obrazy indeksowane, dzięki rozdzieleniu informacji o kolorach i ich przestrzennym rozmieszczeniu w obrazie, pozwalają na zmniejszenie ilości danych obrazowych i umożliwiają ich wyświetlanie za pomocą kart graficznych o małej liczbie dostępnych kolorów. Dokładny opis formatów zapisu obydwu rodzajów obrazów został zamieszczony w rozdziale 1 instrukcji.

Wyświetlanie obrazów kolorowych – różne karty graficzne

W większości komputerów karty graficzne używają 8-, 16- lub 24-bitowej reprezentacji pojedynczego punktu obrazu. Liczba możliwych do wyświetlenia kolorów jest równa $2^{\text{liczba bitów}}$.

Najlepszą jakość obrazu zapewniają systemy wykorzystujące grafikę 24-bitową. Wówczas intensywność każdej z trzech podstawowych barw jest reprezentowana za pomocą ośmiu bitów, czyli można uzyskać 256 poziomów danej barwy. W przypadku grafiki 16-bitowej do reprezentacji kolorów: czerwonego i niebieskiego wykorzystane jest po pięć bitów a kolor zielony może być reprezentowany, w zależności od karty, przez sześć lub, jak pozostałe, przez pięć bitów (jeden bit nie jest wykorzystany). Mamy wówczas 32 lub 64 odcienie danej barwy. Dla kart realizujących grafikę 8-bitową (lub pracujących w trybie 8-bitowym) nie można już rozdzielić bitów odpowiedzialnych za poszczególne barwy podstawowe. Mamy tutaj do czynienia z dwuetapowym tworzeniem obrazu. Wartość zapisana w komórce pamięci reprezentującej wybrany punkt obrazu stanowi adres do tablicy o 24-bitowym słowie, przechowującej informację o wartościach składowych podstawowych dla każdego z 256 kolorów. Ten tryb graficzny nadaje się dobrze do wyświetlania obrazów indeksowanych.

Aby sprawdzić w jakim trybie aktualnie pracuje karta graficzna twojego komputera wprowadź następującą instrukcję:

```
get(0,'ScreenDepth')
```

MATLAB zwróci liczbę całkowitą oznaczającą liczbę bitów reprezentujących pojedynczy punkt na ekranie monitora. Jeżeli zostanie zwrócona wartość 32, to system faktycznie używa grafiki 24-bitowej. Jeżeli karta pracuje w trybie ośmiobitowym, to należy ją przełączyć w tryb pracy 24- lub 16-bitowy. Oczywiście będzie to możliwe jeżeli te tryby są dostępne dla Twojego systemu.

Wykonaj następującą sekwencję poleceń:

```
Im=imread('flowers.tif'); % wczytanie obrazu z pliku
                          % flowers.tif do zmiennej Im
size(Im);                 % sprawdzenie rozmiaru tablicy
                          % reprezentującej obraz,
                          % dla obrazów kolorowych tablica
                          % ta jest trójwymiarowa: m x n x 3
imshow(Im);               % wyświetlenie obrazu
figure, imshow(Im(:,:,1)); % utworzenie nowego rysunku
                          % i wyświetlenie składowej
                          % czerwonej obrazu typu RGB
                          % w postaci obrazu
                          % monochromatycznego
```

Dokonaj podobnego zobrazowania składowych: zielonej i niebieskiej. Zaobserwuj różnice w treści poszczególnych obrazów. Zauważ jak treść obrazu oryginalnego jest powiązana z treścią płaszczyzn kolorów.

Następnie wczytaj obraz indeksowany `'trees.tif'`:

```
[X,map]=imread('trees.tif');
```

oraz spróbuj dokonać takich operacji na palecie kolorów zawartej w zmiennej `map`, aby zobrazować tylko jedną składową koloru jak dla poprzedniego obrazu.

Zmniejszanie liczby kolorów w obrazie

Obrazy typu RGB zawarte w zmiennych klasy `double` zajmują pokaźny obszar pamięci operacyjnej lub pamięci masowej (np. dysku). Na każdy punkt obrazu przypadają 24 bajty pamięci. Obrazy przechowywane w zmiennych klasy `uint8` wymagają trzech bajtów na piksel.
Dla większości obrazów nawet znaczne zmniejszenie liczby występujących kolorów nie pogarsza ich subiektywnie postrzeganej jakości. Jeżeli ograniczymy liczbę kolorów do 256, to możliwe jest zapisanie obrazu jako obrazu indeksowanego przechowywanego w zmiennej klasy `uint8`. Uzyskamy wówczas znaczącą oszczędność pamięci, gdyż każdy piksel jest przechowywany w jednym bajcie. Informacja o obrazie zawarta w mapie kolorów klasy `double` zajmuje zazwyczaj znikomy ułamek objętości danych obrazowych.
Do konwersji obrazu typu RGB na obraz indeksowany służy w środowisku MATLAB funkcja `rgb2ind`. Jej zadaniem jest aproksymowanie kolorów obrazu oryginalnego za pomocą zadanej liczby kolorów lub zestawu kolorów z narzuconej palety oraz konwersja formatu zapisu obrazu. Aproksymowanie kolorów odbywa się poprzez kwantyzację trójwymiarowej przestrzeni koloru, czyli podzielenie tej przestrzeni na takie obszary, że wszystkie punkty obrazu oryginalnego zawarte w jednym obszarze będą reprezentowane przez punkty o jednym kolorze, zazwyczaj ze środka obszaru.

W funkcji `rgb2ind` zaimplementowano dwa sposoby kwantyzacji przestrzeni koloru:

- kwantyzacja równomierna polega na podzieleniu, sześcianu wypełniającego zakres intensywności barw w przestrzeni koloru na mniejsze sześciany, każdy reprezentowany przez jeden kolor ze środka sześcianu,
- metoda najmniejszej wariancji polega na podziale przestrzeni koloru na takie obszary (o dowolnym kształcie) które zapewnią najwierniejsze odtworzenie kolorów występujących najczęściej w obrazie oryginalnym. W efekcie otrzymujemy najmniejsze obszary w miejscach dużego zagęszczenia punktów obrazu oryginalnego w przestrzeni koloru.

Wprowadź sekwencję instrukcji:

```
Im=imread('flowers.tif');
tolerancja=0.2;
[X1,map1]=rgb2ind(Im,tolerancja);
                          % zastosowanie
                          % równomiernej metody kwantyzacji
[X2,map2]=rgb2ind(Im,128);
                          % kwantyzacja do 128 kolorów
                          % metodą najmniejszej wariancji
imshow(Im);               % obrazowanie wyników
figure, imshow(X1,map1);  % przetwarznia
figure, imshow(X2,map2);
```

Zmienna `tolerancja` przyjmująca wartości od 0 do 1 ma znaczenie maksymalnej względnej odległości między punktami w przestrzeni koloru, wewnątrz pojedynczego sześciennego obszaru. Jeżeli drugi argument przyjmuje wartości całkowite większe od jedności, to jego wartość jest interpretowana jako maksymalna liczba kolorów w obrazie wynikowym i stosowana jest metoda najmniejszej wariancji podczas kwantyzacji przestrzeni kolorów.

Sprawdź do jakiej liczby kolorów został skwantowany obraz `X1` i zaobserwuj różnicę jakości pomiędzy obiema metodami. Powtórz powyższe operacje przy założeniu, że obrazy będą skwantowane do 32 kolorów i porównaj wyniki.

Wywołanie funkcji `rgb2ind` z drugim argumentem będącym paletą kolorów narzuci tę paletę w obrazie wynikowym. Będzie miał wówczas miejsce proces mapowania kolorów do aktualnej palety. Użycie tej samej palety barw do różnych obrazów jest korzystne z punktu widzenia ich jednoczesnego wyświetlania w trybie grafiki 8-bitowej. Jednakże stosowanie jednakowej palety do obrazów o odmiennej treści powoduje znaczne pogorszenie ich jakości w stosunku do palet indywidualnych dla danych obrazów. Można się o tym przekonać wykonując następujące instrukcje:

```
RGB1=imread('autumn.tif');
imshow(RGB1);
RGB2=imread('flowers.tif');
[X1,map]=rgb2ind(RGB1,100);  % ograniczenie liczby kolorów
                             % do 100 i utworzenie palety
figure, imshow(X1,map);
X2=rgb2ind(RGB2,map);        % utworzenie obrazu indeksowanego
                             % z zadaną mapą kolorów
figure, imshow(X2,map);
```

Do zmniejszenia liczby kolorów w obrazie indeksowanym służy funkcja `imapprox`, która jest sekwencją funkcji `ind2rgb` i `rgb2ind` z zadaną maksymalną liczbą kolorów. Sekwencja operacji:

```
load trees;
[Y,newmap]=imapprox(X,map,64);
```

spowoduje zmniejszenie liczby kolorów w obrazie indeksowanym do 64.

Dla polepszenia subiektywnego odbioru koloru w obrazach funkcje `rgb2ind` i `imapprox` dokonują operacji określanej w języku angielskim mianem: „dithering". Polega ona na takiej modyfikacji kolorów pikseli w sąsiedztwie danego punktu obrazu aby uśredniony kolor był jak najbliższy temu w obrazie oryginalnym. Uzyskujemy wówczas poprawę jakości koloru kosztem rozdzielczości obrazu. Aby zablokować tę operację należy wywołać funkcję z argumentem `'nodither'`, np:

```
[X,map]=rgb2ind(RGB,100,'nodither');
```

Biblioteka IPT pakietu MATLAB zapewnia również możliwość konwersji obrazów zapisanych w formacie RGB do innych przestrzeni koloru, w wyniku której otrzymujemy następujące formaty zapisu:

- format NTSC,
- format HSV.

Zapoznaj się z funkcjami: `rgb2ntsc` i `rgb2hsv` realizującymi powyższe konwersje.

Operacje punktowe na obrazach

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z pojęciem histogramu obrazu, histogramu skumulowanego oraz z procedurami IPT pakietu MATLAB dotyczącymi przetwarzania obrazów za pomocą operacji punktowych oraz modelowania ich histogramów.

Funkcja `impixel`

Funkcja ta zwraca wartość zaznaczonego elementu lub elementów obrazu. Współrzędne elementu obrazu można podać jako argument funkcji lub zaznaczyć dany element z wykorzystaniem myszy. W tym drugim przypadku, po wywołaniu funkcji i najechaniu kursorem na obraz, kursor przybierze kształt krzyża. Po zaznaczeniu elementów obrazu, których wartości chcemy poznać, należy wcisnąć `<Enter>`, wtedy zostaną wyświetlone wartości zaznaczonych punktów obrazu.

Niezależnie od typu obrazu, funkcja `impixel` zawsze zwraca trzy wartości RGB:

- dla obrazów typu RGB, funkcja zwraca aktualną wartość zaznaczonego elementu, typu `uint8` lub `double`,
- dla obrazów indeksowanych, funkcja zwraca trójkę liczb przechowywanych w rzędzie macierzy będącej mapą kolorów danego obrazu. Numer rzędu tej macierzy jest określany przez wartość zaznaczonego elementu obrazu,
- dla obrazów monochromatycznych, funkcja zwraca trzy jednakowe liczby (R=G=B), typu `uint8` lub `double`.

Załaduj i wyświetl obraz `cameraman.tif` oraz wykorzystując funkcję `impixel` określ wartości kilku wybranych elementów obrazu.

```
I=imread('cameraman.tif');
imshow(I);
impixel       % zaznacz za pomocą myszy kilka elementów
              % obrazu + <Enter>
```

Funkcja `improfile`

Funkcja ta pozwala na uzyskanie rozkładu wartości elementów obrazu wzdłuż zadanego odcinka lub łamanej składającej się z kilku odcinków. Współrzędne odcinków można podać jako argumenty funkcji, lub do ich zaznaczenia można wykorzystać mysz. Dla pojedynczego odcinka, funkcja zwraca dwuwymiarowy rozkład wartości elementów obrazu, w przypadku kilku zaznaczonych odcinków otrzymuje się trójwymiarowy rozkład wartości. W przypadku zaznaczania odcinków za pomocą myszy, po najechaniu kursorem na obraz przybierze on kształt krzyża, wciśnięcie lewego przycisku myszy zaznacza początek odcinka, wciśnięcie prawego przycisku – jego koniec. Jeżeli chcemy zaznaczyć łamaną, powtórne wciśnięcie lewego przycisku myszy (po wybraniu początkowego punktu pierwszego odcinka) określa koniec pierwszego odcinka i początek drugiego. Procedurę należy powtarzać aż do uzyskania żądanej łamanej.

Wykorzystaj funkcję `improfile` dla określania profilu rozkładu jasności elementów obrazu wzdłuż wybranego odcinka:

```
improfile
```

Funkcja `imcontour`

Funkcja ta pozwala na wyświetlenie poziomic dla danego obrazu o jednakowych wartościach jasności. Liczba poziomic jest jednym z argumentów funkcji:

```
imcontour(nazwa_obrazu, liczba_poziomic)
```

Brak argumentu określającego liczbę poziomic powoduje automatyczne określenie ich liczby. Wartość poziomic można odczytać za pomocą funkcji `clabel`, do tego celu należy jednak wyznaczyć dodatkowe macierze `cs` i `h` generowane przez `imcontour:`

```
[cs, h] = imcontour(I, 2)
clabel(cs, h);
```

Histogram i histogram skumulowany

Polepszanie jakości obrazów, w tym poprawa jakości obrazów i segmentacja należą do podstawowych metod wstępnego przetwarzania obrazów. Jedną z takich metod jest metoda modelowania histogramu, która jest oparta na statystycznej analizie przetwarzanego obrazu. Histogramem obrazu cyfrowego o rozmiarach M na N oraz liczbie poziomów jasności G nazywamy wektor His, którego elementom His(g), g = 0,...,G-1 odpowiada liczba punktów obrazu o jasnościach g. Histogram wykorzystywany jest m. in. do ustalania optymalnej wartości progu przy segmentacji obrazów. Segmentacja przez progowanie polega na podziale obrazu na dwie klasy. Punktom obrazu o jasnościach z przedziału $< 0...g_{th})$ przypisywana jest jasność 0 a punktom z przedziału $<g_{th} ... G-1>$ - jasność 1. Wartość g_{th} jest arbitralnie wybraną wartością progu. Utworzony w ten sposób obraz **binarny** posiada tylko dwa poziomy jasności.

Wyświetl histogram załadowanego obrazu za pomocą sekwencji procedur:

```
figure, imhist(I)
```

Ponadto poprzez korekcję histogramu dokonuje się poprawy jakości obrazu np. w przypadku jego prześwietlenia lub niedoświetlenia. Skutkiem złych warunków oświetlenia jest nieefektywne wykorzystanie wszystkich poziomów kwantowania co powoduje, że niektóre elementy histogramu His(g) przyjmują wartość równą zeru. Korekcja histogramu polega na takim przekształceniu jego elementów aby histogram wynikowy był maksymalnie płaski i równomiernie wypełniał cały zakres jasności <0...G-1>. Taki sposób korekcji nazywamy **wyrównywaniem histogramu**. Jednym ze sposobów korekcji histogramu jest przyporządkowanie każdemu punktowi obrazu jasności proporcjonalnej do wartości histogramu skumulowanego His_{sk} dla poziomu g

$$His_{sk}(g) = \frac{1}{MN}\sum_{k=0}^{g} His(k)$$

$$m(g) = His_{sk}(g)(G-1)$$

gdzie m(g) – oznacza nową jasność punktu obrazu o współrzędnych (i,j) po korekcji histogramu.

Zastosuj procedurę wyrównywania histogramu do obrazu `cameraman.tif`:

```
J=histeq(I);
figure, imhist(J)
figure, imshow(J)
```

Porównaj obraz oryginalny i przetworzony oraz ich histogramy

Liniowe przekształcenie poziomów jasności obrazu

Metoda ta polega na zmianie odwzorowania poziomów jasności analizowanego obrazu. Ilustruje to rys. 1. Na osi *0x* pokazane są poziomy jasności obrazu oryginalnego. Jasności pomiędzy wartościami g_{min} i g_{max} zostają liniowo odwzorowane na nowy zakres jasności obrazu wynikowego, określony wartościami h_{min} oraz h_{max}. W wyniku działania procedury, przy odpowiednio dobranych wartościach progów g_{min}, g_{max}, h_{min}, h_{max} można uzyskać poprawę kontrastu obrazu. W bibliotece IPT do modyfikacji odwzorowania poziomów jasności obrazu służy funkcja `imadjust`:

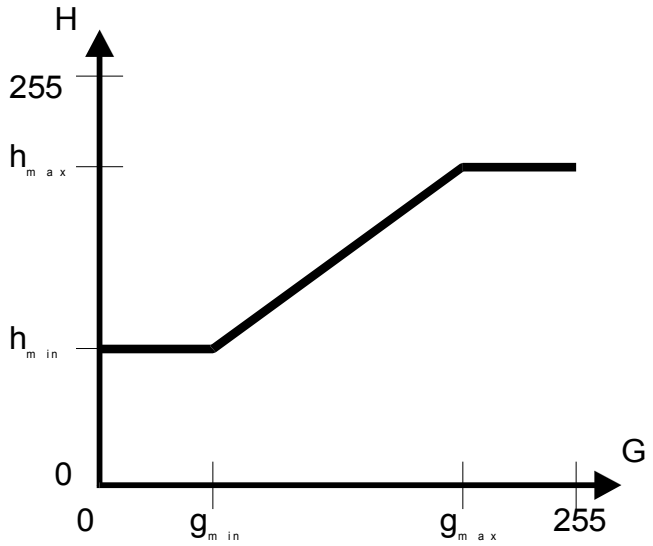
```
imadjust(I, [gmin gmax], [hmin hmax]);
```

Uwaga: poziomy progów zawsze przyjmują wartości z zakresu [0, 1], niezależnie od typu przetwarzanego obrazu (`uint8` lub `double`).

Zastosuj procedurę modyfikacji poziomów jasności dla załadowanego uprzednio obrazu:

```
J=imadjust(I, [0.05 0.8], [0 1]);
figure, imshow(J)
```

Jeżeli wartości h_{min} i h_{max} wynoszą odpowiednio 0 i 1, to taka metoda poprawy jakości obrazu określana jest jako "rozciąganie histogramu" (ang. *stretching*).



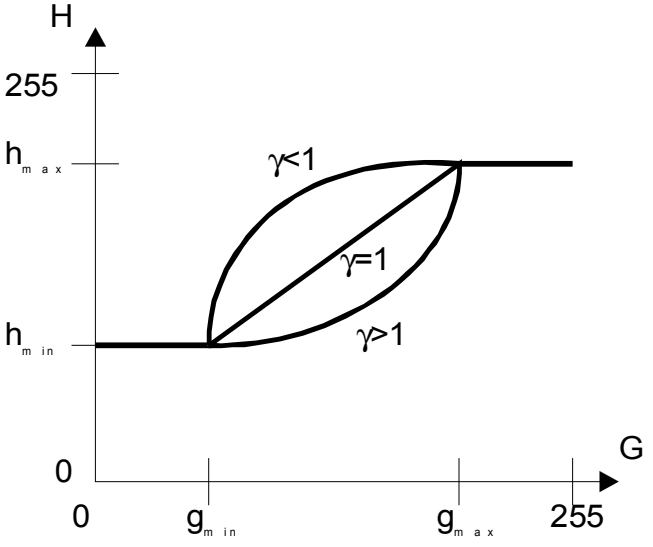
Rys. 1 Liniowe przekształcenie poziomów jasności obrazu

Nieliniowe przekształcenie poziomów jasności obrazu

Funkcja `imadjust` pozwala również na nieliniowe odwzorowanie poziomów jasności obrazu. Do tego celu służy dodatkowy parametr γ:

```
imadjust(I, [gmin gmax], [hmin hmax], γ);
```

Parametr γ przyjmuje wartości z zakresu $[0, \infty)$. Dla $\gamma<1$ w obrazie wynikowym są uwypuklane poziomy jasności o większych wartościach (jaśniejsze), zaś dla $\gamma>1$ – poziomy o mniejszych wartościach (ciemniejsze). W przypadku $\gamma=1$ odwzorowanie jest odcinkowo liniowe. Ilustruje to rys. 2.



Rys. 2 Nieliniowe przekształcenie poziomów jasności obrazu

Za pomocą funkcji `imadjust` wykonaj nieliniowe odwzorowanie poziomów jasności dla obrazu `cameraman.tif` dla γ=0.5 oraz γ=3.5:

```
J=imadjust(I, [ ], [ ], γ);
```

Porównaj otrzymane obrazy oraz ich histogramy z obrazem oryginalnym i jego histogramem.

Z działaniem omówionych funkcji modelowania histogramu obrazu możesz również się zapoznać korzystając z procedur demonstracyjnych pakietu MATLAB. Po wprowadzeniu polecenia `demo` wybierz Toolboxes -> Image Processing -> Intensity Adjustement and Histogram Equalisation.

Filtracja w dziedzinie przestrzennej obrazu

Cel ćwiczenia

Celem ćwiczenia jest poznanie funkcji pakietu MATLAB związanych z wykonywaniem dwuwymiarowego splotu oraz filtracji obrazów w dziedzinie przestrzennej. Pokazane również zostaną wybrane sposoby usuwania zakłóceń z obrazów cyfrowych za pomocą filtracji nieliniowej.

Definicja filtracji w dziedzinie przestrzennej

Filtry liniowe i nieliniowe są szeroko stosowane we wstępnych metodach przetwarzania obrazów. Należą do metod poprawy jakości obrazu, gdyż w przypadku ich stosowania nie stosuje się kryteriów analitycznych określających poprawę jakości lecz kryteria subiektywne.

Filtracja w dziedzinie przestrzennej jest zdefiniowana następująco:

$$G(i,j) = H(i,j)**f(i,j)$$

gdzie H(i,j) – obraz oryginalny, G(i,j) – obraz po filtracji, f(i,j) – funkcja określająca filtr. Operator ** oznacza dwuwymiarowy splot, który dla dyskretnych, okresowych funkcji H(i,j) i f(i,j) o okresach (N_1, N_1) i (N_2, N_2) odpowiednio, jest określony następująco:

$$H_e(i,j) * *f_e(i,j) = \sum_{m=0}^{M-1}\sum_{n=0}^{M-1} H_e(m,n) f_e(i-m, j-n)$$

gdzie

$$H_e(i,j) = \begin{cases} H(i,j) & \mathbf{0} \le i,j \le N_1 \\ \mathbf{0} & N_1 \le i,j \le M \end{cases} \qquad f_e(i,j) = \begin{cases} f(i,j) & \mathbf{0} \le i,j \le N_2 \\ \mathbf{0} & N_2 \le i,j \le M \end{cases}$$

funkcje f_e oraz H_e posiadają sztucznie rozszerzone okresy do wartości $M=N_1+N_2-1$.

Procedury do realizacji filtracji w pakiecie MATLAB

W bibliotece IPT dwuwymiarowy splot realizuje funkcja `conv2(H, f)`, gdzie `H` – tablica zawierająca analizowany obraz, `f` – tablica definiująca filtr. Operacja splotu jest realizowana przez funkcję `conv2` w następujących etapach:

- na podstawie tablicy określającej filtr `f` jest wyznaczana nowa tablica za pomocą funkcji `rot90`. Funkcja ta powoduje rotację elementów tablicy filtru o krotność 90°, np.

```
f = [1 2 3; 4 5 6]
f1 = rot90(f, 2) = [6 5 4;3 2 1]
```

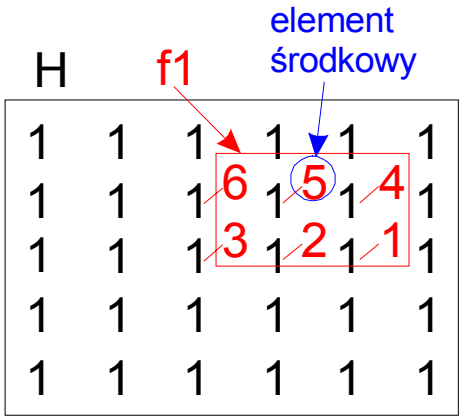
(dodatkowy argument funkcji `rot90` informuje, o jaką krotność 90° dokonać rotacji elementów tablicy, w tym przypadku o 2*90°=180°)

- następnie wyznaczane są współrzędne elementu środkowego tablicy `f1`, określone jako

```
w = floor((size(f1)-1)/2);
```

gdzie `w` – wektor zawierający współrzędne elementu środkowego tablicy `f1`, `floor` – funkcja zaokrąglająca wartość argumentu do najbliższej liczby całkowitej ale w kierunku do -∞, np. `floor(0.9)=0`, `size` – funkcja zwracająca liczbę wierszy i kolumn tablicy będącej jej argumentem.

- następnie tablica `f1` przesuwana jest względem analizowanego obrazu `H` w taki sposób, że element środkowy `f1` ustawiany jest nad każdym elementem obrazu; następnie obliczana jest wartość sumy iloczynów elementów `f1` oraz odpowiadających im punktów obrazu. Ilustruje to rys. 1.



Rys. 1 Przykład wykonywania operacji splotu

Dla przykładu pokazanego na rys. 1, wynik splotu dla elementu obrazu wynikowego `G` o współrzędnych (4, 2) wynosi

```
G(4, 2) = 1*6+1*5+1*4+1*3+1*2+1*1 = 21
```

Wykonaj dwuwymiarowy splot dla tablicy `H` oraz filtru określonego poprzez `f` wykonując następującą sekwencję komend:

```
f =[1 2 3; 4 5 6];
H=ones(5, 6);
G=conv2(H, f);
```

Zwróć uwagę na wymiary tablicy `G`.

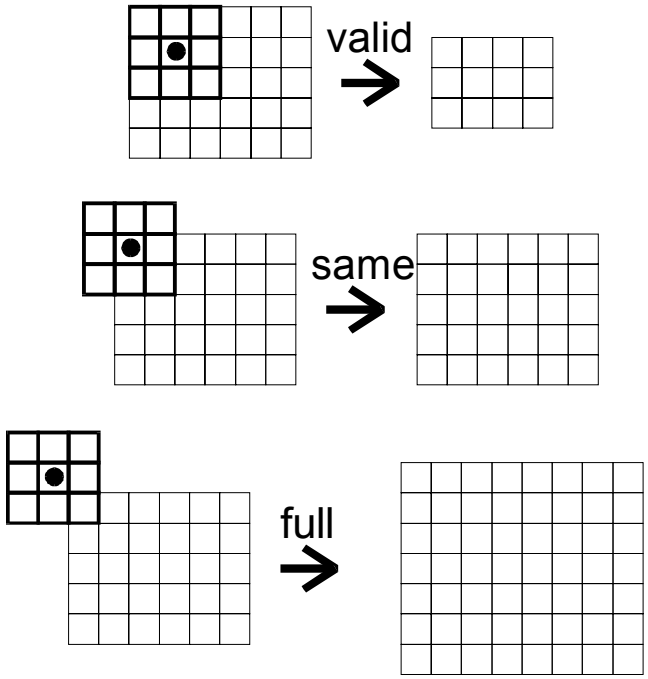
Ponieważ podczas operacji splotu okresy funkcji są zwiększane poprzez dodanie elementów równych zeru, obraz wynikowy ma zawsze większy rozmiar niż obraz źródłowy. Aby uniknąć problemów związanych ze zwiększeniem wymiarów obrazu funkcja `conv2` posiada dodatkowy argument pozwalający na ograniczenie zasięgu operacji splotu. Wartości tego argumentu oraz ich znaczenie są następujące:

- `'valid'` – splot wyznaczany jest tylko dla tych elementów obrazu, w których otoczeniu nie ma elementów uzupełnianych zerami. W tym przypadku wymiar obrazu wynikowego

jest mniejszy o liczbę kolumn i wierszy zależną od wielkości filtru. Np. dla obrazu o wymiarach 5×6 elementów i filtru 3×3 obraz wynikowy będzie miał wymiary 3×4.

- `'same'` – splot wyznaczany jest w taki sposób, że wymiary obrazów źródłowego i wynikowego są takie same. W zależności od wymiarów filtru, część elementów obrazu wynikowego jest wyznaczona na podstawie uzupełnienia zerami obrazu źródłowego. Np. dla obrazu o wymiarach 5×6 elementów i filtru 3×3 obraz wynikowy będzie miał wymiary 5×6, przy czy elementy pierwszej kolumny i pierwszego rzędu oraz ostatniej kolumny i ostatniego rzędu będą wyznaczone na podstawie uzupełnienia zerami obrazu źródłowego.
- `'full'` – splot obliczany jest dla wszystkich elementów obrazu, jeżeli dowolny element tablicy filtru (niekoniecznie środkowy) pokrywa się z punktem obrazu. Zatem wymiar obrazu wynikowego jest większy niż wymiar obrazu źródłowego. Np. dla obrazu o wymiarach 5×6 elementów i filtru 3×3 obraz wynikowy będzie miał wymiary 7×8, przy czy elementy pierwszej i drugiej kolumny, pierwszego i drugiego rzędu oraz ostatniej i przedostatniej kolumny, ostatniego i przedostatniego rzędu będą wyznaczone na podstawie uzupełnienia zerami obrazu źródłowego.

Efekty wpływu tego parametru na wymiar obrazu wynikowego pokazuje rys. 2.



Rys. 2 Wymiary obrazu wynikowego przy różnych parametrach funkcji `conv2` dla obrazu źródłowego o wymiarach 5×6 i filtru o wymiarach 3×3.

Inną funkcją umożliwiającą wykonanie filtracji w dziedzinie przestrzennej jest `filter2`. Różni się ona od `conv2` tym, że tablica definiująca filtr nie podlega rotacji o 180° (nie jest wykonywana funkcja `rot90`). W konsekwencji, funkcja `filter2` wykonuje operację korelacji. Jednak, w większości zastosowań przetwarzania obrazów tablice określające filtry są symetryczne względem obydwu głównych przekątnych, co powoduje że operacje splotu i korelacji są równoważne. Składnia funkcji `filter2` jest następująca:

```
G=filter2(f, H, parametr);
```

gdzie `H`, `f` – tablice określające odpowiednio obraz źródłowy i filtr, zaś `parametr` ={`'valid'`, `'same'`, `'full'`}. Znaczenie tych parametrów jest takie samo jak dla `conv2`.

Dla filtru

```
f2=[1 2 1;2 4 2;1 2 1];
```

wykonaj operację splotu i filtracji dla tablicy `H`:

```
G1=conv2(H, f2, 'valid');
G2=filter2(f2, H, 'valid');
```

porównaj otrzymane rezultaty.

Bardzo użyteczną funkcją pakietu MATLAB jest funkcja `fspecial`. Pozwala ona na określenie predefiniowanych filtrów, najczęściej stosowanych w metodach analizy obrazów. Jej składnia jest następująca:

```
f=fspecial('nazwa_filtru', n);
```

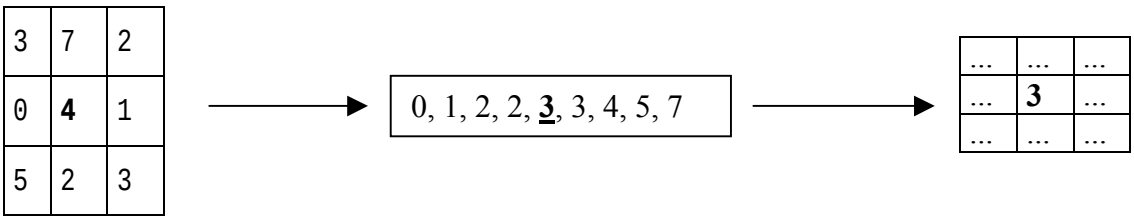
gdzie `'nazwa_filtru'` określa jeden z predefiniowanych filtrów zaś `n` jest wymiarem tablicy filtru.
Zapoznaj się z opisem funkcji `fspecial` w celu poznania predefiniowanych filtrów występujących w pakiecie MATLAB. Wykonaj przykładową filtrację obrazu `'cameraman.tif'` za pomocą filtru uśredniającego o wymiarach 5×5 wykonując następującą sekwencją komend:

```
H=imread('cameraman.tif');
imshow(H);
f=fspecial('average',5);
G=filter2(f, H);
figure, imshow(G);
```

Dlaczego nie udało się wyświetlić obrazu wynikowego? Zaproponuj rozwiązanie tego problemu.
Kiedy uda Ci się wyświetlić obraz wynikowy, dokonaj filtracji obrazu za pomocą różnych filtrów określonych przez `fspecial`. Porównaj uzyskane rezultaty.

Filtracja medianowa

Filtr medianowy pozwala na usuwanie zakłóceń w obrazach o charakterze impulsowym, tzn. takich, których amplituda jest zbliżona do wartości minimalnych lub maksymalnych. W procesie filtracji każdy punkt obrazu wynikowego jest środkowym elementem uporządkowanego pod względem wartości ciągu pikseli odpowiadającego mu punktu obrazu źródłowego oraz jego sąsiedztwa, którego wielkość zależy od rzędu filtru.



Rys.3 Przykład wykonywania filtracji medianowej

Na rys. 3 pokazano przykład wykonywania filtracji medianowej dla punktu o jasności 4 przy założeniu, że analizowane sąsiedztwo ma o wymiary 3x3. Elementy obrazu należące do tego sąsiedztwa są sortowane zgodnie z narastającą wartością, a następnie jest wybierany element środkowy jako punkt obrazu wynikowego. Wielkość sąsiedztwa decyduje o „intensywności" filtracji.

W pakiecie MATLAB filtracja medianowa jest realizowana za pomocą funkcji `medfilt2`. Dla celów porównania skuteczności różnych rodzajów filtrów została zaimplementowana funkcja `imnoise`, generująca zakłócenia o zadanym rozkładzie.

Poniższy przykład obrazuje skuteczność filtracji zakłócenia typu „*salt and pepper*" powstałego przez zastąpienie wartości niektórych pikseli największą lub najmniejszą możliwą wartością.

```
I=imread('eight.tif');              % wczytanie obrazu
J=imnose(I,'salt & pepper',0.02);   %dodanie zniekształceń
imshow(I);
figure, imshow(J);
L=medfilt2(J,[3 3]);                % filtracja medianowa
                                    % z sąsiedztwem 3x3
figure, imshow(L);
```

Trzeci parametr procedury `imnoise` wskazuje jaki procent punktów obrazu ma ulec zakłóceniom. Przeprowadź powyższe obliczenia także dla obrazu z większą liczbą punktów poddanych zakłóceniu oraz większego rozmiaru sąsiedztwa.

Transformacja Fouriera obrazu

Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z procedurami biblioteki IPT programu MATLAB służącymi do wyznaczania dyskretnej transformacji Fouriera obrazu oraz projektowania filtrów w dziedzinie częstotliwości.

Definicja DFT

Dyskretna transformacja Fouriera dla przypadku dwuwymiarowego, prosta i odwrotna, zdefiniowana jest za pomocą następującej pary wzorów:

$$F(u,v) = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} f(i,j) e^{-j(2\pi/M)u} e^{-j(2\pi/N)v} \qquad i = 0,1,...,M-1 \qquad j = 0,1,...,N-1$$

$$f(i,j) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v) e^{j(2\pi/M)u} e^{j(2\pi/N)v} \qquad u = 0,1,...,M-1 \qquad v = 0,1,...,N-1$$

gdzie f(i,j) – element obrazu o współrzędnych (i,j), F(u,v) – transformata Fouriera tego punktu zaś M, N – wymiary obrazu.

W praktyce, do wyznaczania transformacji Fouriera obrazu wykorzystuje się algorytm szybkiej transformacji Fouriera (ang. *Fast Fourier Transform, FFT*) co pozwala na znaczne ograniczenie liczby wykonywanych operacji w konsekwencji powodując skrócenie czasu obliczeń.

W przypadku przetwarzania obrazów do filtracji najczęściej wykorzystuje się filtry o skończonej odpowiedzi impulsowej (ang. *Finite Impulse Response, FIR*). Jest to spowodowane m.in. możliwością reprezentacji transmitancji tych filtrów w postaci macierzowej, istnieniem szeregu efektywnych metod projektowania tych filtrów oraz możliwością uzyskania liniowej fazy filtru, co powoduje ograniczenie zniekształceń podczas procesu filtracji. W ćwiczeniu zostaną pokazane trzy metody projektowania filtrów FIR: metoda próbkowania w dziedzinie częstotliwości, metoda okien oraz metoda tworzenia macierzy transmitancji filtru.

Wyznaczanie DFT obrazu

załaduj obraz `cameraman.tif` z dysku:

```
I=imread('cameraman.tif','tif');
```

wykonaj DFT dla załadowanego obrazu:

```
F=fft(I);
```

wyświetl widmo amplitudowe uzyskanej transformaty obrazu:

```
F1=log(abs(F)+1);
```

```
F1=F1/max(max(F1));
imshow(F1);
```

składowa stała (współczynnik transformaty Fouriera o współrzędnych (0,0) znajduje się lewym górnym rogu obrazu. Przeważnie, dla lepszej wizualizacji widma amplitudowego, składowa stała powinna znajdować się w centrum obrazu widma. Dla uzyskania tego efektu służy funkcja `fftshift`, dokonująca przesunięcia obrazu widma o połowę okresu, w kierunku osi *u* i *v* :

```
F1= fftshift(log(abs(F)+1);
F1=F1/max(max(F1));
imshow(F1);
```

Projektowanie filtrów w dziedzinie częstotliwości

Metoda próbkowania w dziedzinie częstotliwości

W przypadku tej metody transmitancja filtru wytwarzana jest na podstawie macierzy opisującej charakterystykę odpowiedzi impulsowej filtru. Transmitancja filtru w dziedzinie częstotliwości przechodzi przez punkty zdefiniowane przez tą macierz. Próbkowanie w dziedzinie częstotliwości nie określa zachowania się transmitancji filtru poza punktami macierzy, mogą tam wystąpić oscylacje.

Poniższy przykład tworzy transmitancję dolnoprzepustowego filtru `h1` na podstawie macierzy `Hd` charakteryzującej jego odpowiedź impulsową o wymiarach 11×11:
zdefiniuj macierz `Hd`

```
[m,n]=Size(I);
Hd=zeros(m,n); Hd(80:160,80:160)=1;
```

zdefiniuj układ współrzednych w dziedzinie częstotliwości:

```
[f1,f2]=freqspace(m,'meshgrid');
```

wyświetl macierz `Hd`

```
mesh(f1,f2,Hd); axis([-1 1 –1 1 0 1.2]); colormap(jet(64));
```

wyznacz transmitancję filtru

```
h=fsamp2(Hd);
```

wyświetl transmitancję filtru w dziedzinie częstotliwości

```
figure; h1=freqz2(h,[m,n]); axis([-1 1 –1 1 0 1.2]);
```

Można zauważyć oscylacje dla punktów transmitancji filtru leżącymi poza punktami macierzy `Hd`, która ją charakteryzuje. Jest to główna wada tej metody.

Metoda okien

Podobnie jak w przypadku filtrów jednowymiarowych, ograniczenie oscylacji transmitancji filtru można uzyskać poprzez pomnożenie idealnej odpowiedzi filtru przez określoną funkcję okna. Ilustruje to poniższy przykład:
dla zdefiniowanej uprzednio macierzy `Hd` opisującej filtr wyznacz jego transmitancję wykorzystując okno Hamminga:

```
h=fwind1(Hd,hamming(m));
figure; h2=freqz2(h,[m,n]); axis([-1 1 –1 1 0 1.2]);
```

Otrzymana transmitancja filtru nie zawiera oscylacji jak w przypadku transmitancji `h1`.

Metoda tworzenia macierzy transmitancji filtru

W metodzie tej tworzy się bezpośrednio macierz opisującą transmitancję filtru w dziedzinie częstotliwości. Do tego celu służy funkcja `freqspace` która tworzy odpowiedni układ współrzędnych w dziedzinie częstotliwości dla dowolnej wielkości macierzy transmitancji, znormalizowany do zakresu częstotliwości [-1, 1], co odpowiada zakresowi częstotliwości z przedziału $[-\pi,\pi]$.

Przykład pokazuje przykład utworzenia transmitancji filtru dolnoprzepustowego o częstotliwości granicznej równej 0,5 ($\pi/2$).

zdefiniuj układ współrzędnych w dziedzinie częstotliwości:

```
[f1,f2] =freqspace(m,'meshgrid');
```

oraz transmitancję filtru

```
h3=zeros(m,n); d=sqrt(f1.^2+ f2.^2)<0.5;
h3(d)=1;
```

wyświetl transmitancję filtru `h3`:

```
figure; mesh(f1,f2,h3)
```

Filtracja w dziedzinie częstotliwości

Filtracja w dziedzinie częstotliwości określona jest następującym wzorem:

$$g(i,j) = \mathrm{IFFT}[F(u,v)H(u,v)]$$

gdzie g(i,j) – obraz po filtracji, H(u,v) - transmitancja filtru w dziedzinie częstotliwości.

Dokonaj filtracji załadowanego obrazu `cameraman.tif` za pomocą utworzonych poprzednio macierzy transmitancji `h1`, `h2` i `h3`:

```
G=ifft2(fftshift(h1).*fft2(I));
```

`fftshift(h1)` powoduje przesunięcie transmitancji filtru o pół okresu, tak aby środek układu współrzędnych znalazł się w lewym górnym rogu macierzy.

Wyświetl obraz

```
G1=abs(G)/max(max(abs(G)));
imshow(G1)
```

Operację filtracji i wyświetlania uzyskanego obrazu powtórz dla transmitancji `h2` i `h3`.

Metody kompresji obrazów

Cel ćwiczenia

Celem ćwiczenia jest poznanie metody kompresji obrazów cyfrowych wykorzystującej dwuwymiarową dyskretną transformację kosinusową (DCT). Ćwiczenie bazuje na wykorzystaniu funkcji biblioteki *Image Processing Toolbox* pakietu MATLAB.

Redundancja danych obrazowych

Obrazy wizualne, reprezentowane w postaci dwuwymiarowej tablicy jasności, charakteryzują się zazwyczaj dużą *redundancją*, tj. nadmierną liczbą danych, wykorzystywanych do kodowania obrazu. Zadaniem metod *kompresji obrazów* jest redukcja liczby danych koniecznych do reprezentacji informacji obrazowej. Metody te bazują na eliminacji redundancji zawartej w obrazie.

Wyróżniamy trzy rodzaje redundancji obrazów wizualnych:

- redundancja kodowania,
- przestrzenna,
- psychowizualna.

Redundancja kodowania wiąże się ze sposobem kodowania jasności poszczególnych pikseli. Najczęściej pojedynczy piksel jest opisany słowem o stałej liczbie bitów. W przypadku, kiedy częstości występowania w obrazie poszczególnych jasności różnią się znacznie między sobą, ten sposób kodowania danych obrazowych jest nieefektywny. Można wówczas dokonać kompresji obrazu przez zastosowanie kodu o zmiennej długości słowa (liczbie bitów). Jasnościom występującym najczęściej przypisuje się kody o najmniejszej długości, jasnościom występującym najrzadziej - najdłuższe. Optymalne rezultaty kompresji uzyskuje się stosując metodę Huffmana kodowania obrazów.
Redundancja przestrzenna (ang. *interpixel redundancy*) związana jest z korelacją jasności punktów obrazu położonych blisko siebie. Cechą charakterystyczną większości obrazów jest występowanie relatywnie dużych powierzchni o małych różnicach jasności w ich obrębie. Fakt ten pozwala przewidzieć z dużym prawopodobieństwem jasność danego piksela na podstawie punktów obrazu z jego sąsiedztwa.
Trzeci rodzaj redundancji wynika z fizjologicznych własności oka. *Redundancję psychowizualną* usuwa się przez kwantowanie ciągłej funkcji jasności obserwowanej sceny. Prawidłowo wykonane kwantowanie nie pogarsza subiektywnego odbioru obrazu, mimo faktycznej straty części informacji.

Metody kompresji obrazów dzielimy na dwie podstawowe grupy:

- metody bezstratne,
- metody stratne.

Metody bezstratne charakteryzują się możliwością wiernego odtworzenia obrazu ze zbioru danych po kompresji. W przypadku metod stratnych część informacji jest tracona na rzecz poprawy współczynnika kompresji danych.

Dla bardzo szerokiej klasy obrazów dominującą przyczyną nadmiarowości danych jest redundancja przestrzenna, której usuwanie jest relatywnie łatwe w stratnych metodach kom-

presji, bazujących na transformacjach obrazu. Wobec tego skupimy uwagę na transformacji DCT i możliwości dokonania kompresji danych otrzymanych w jej wyniku.

Definicja i właściwości dwuwymiarowej dyskretnej transformacji kosinusowej

Dwuwymiarowa transformacja DCT macierzy A wymiarach M×N, reprezentującej np. obraz cyfrowy, dana jest zależnością:

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad 0 \le p \le M-1,\ 0 \le q \le N-1$$

gdzie:

$$\alpha_p = \begin{cases} 1/\sqrt{M}, & p = 0 \\ \sqrt{2/M}, & 1 \le p \le M-1 \end{cases} \qquad \alpha_q = \begin{cases} 1/\sqrt{N}, & q = 0 \\ \sqrt{2/N}, & 1 \le q \le N-1 \end{cases}$$

Transformacja DCT jest odwracalna dzięki czemu współczynniki macierzy A można otrzymać z jej transformaty B:

$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}, \quad 0 \le m \le M-1,\ 0 \le n \le N-1$$

gdzie α_p i α_q są definiowane jak poprzednio.

W środowisku MATLAB powyższe operacje są wykonywane za pomocą funkcji `dct2` oraz `idct2`. Różnicą w stosunku do podanej definicji jest indeksowanie współczynników. Indeksy tablic w MATLABie zaczynają się od jedynki.

```
A=[1 3;2 7]          % utworzenie przykładowej macierzy
B=dct2(A)            % obliczenie jej dwuwymiarowej DCT
C=idct2(B)           % oraz odwrotnej DCT
```

Elementy macierzy C powinny przyjąć takie same wartości jak odpowiednie elementy macierzy A.

Dla szybkiego otrzymania DCT kwadratowych macierzy o niewielkich rozmiarach zaimplementowano efektywną metodę jej obliczania, bazującą na tzw. macierzy transformacji DCT zawierającej wartości funkcji bazowych jednowymiarowej DCT. Macierz transformacji tworzy się wywołując funkcję:

```
T=dctmtx(rozmiar)
```

Wówczas dwuwymiarową DCT macierzy A otrzymujemy następująco: B=T*A*T′ a transformata odwrotna jest dana zależnością: A=T′*B*T.

Powtórz przeprowadzone poprzednio obliczenie z wykorzystaniem zdefiniowanej macierzy:

```
T=dctmtx(2)          % utworzenie macierzy transformacji
B2=T*A*T'            % obliczenie dwuwymiarowej DCT
C2=T'*B2*T           % oraz transformaty odwrotnej
```

Transformacja DCT jest postawą algorytmu kompresji JPEG. Obraz w tym algorytmie jest dzielony na małe bloki, 8x8 lub 16x16 pikseli i dokonywana jest DCT każdego bloku. Otrzymane współczynniki DCT są następnie kwantowane, kodowane i przesyłane do odbiornika lub zapisywane do pliku. W odbiorniku lub programie rozpakowującym dokonywane są operacje odwrotne. Dla szerokiej klasy obrazów większość współczynników DCT przyjmuje wartości bliskie zeru, wobec czego można je pominąć w procesie kodowania i zastąpić zerami podczas rekonstrukcji obrazu bez znaczącego pogorszenia jego jakości.

Wykonaj następującą sekwencję operacji i zaobserwuj rozkład współczynników DCT:

```
I=imread('pout.tif');
A=double(I(1:16,1:16))/255;    %konwersja klasy liczb
                               %i wycięcie fragmentu obrazu
T=dctmtx(16);          % utworzenie macierzy transformacji
B=T*A*T';              % obliczenie dwuwymiarowej DCT
surf(B);               % zobrazowanie wartości współczynników
```

Do dalszych obliczeń pomocna będzie funkcja przetwarzania obrazu w blokach:

```
B = blkproc(A,[M N],fun,P1,P2,...)
```

gdzie `A` jest przetwarzanym obrazem podzielonym wcześniej na bloki o rozmiarach M×N. Funkcja `fun` oddziałuje na każdy blok obrazu:

```
Y=fun(X)
```

i może być wyrażeniem, tekstem zawierającym nazwę funkcji lub funkcją typu inline. `P1, P2,...` są opcjonalnymi parametrami funkcji `fun`.

Wykonanie poniższego przykładu, w którym usunięto 54 z 64 współczynników DCT pokaże przydatność tej transformacji w kompresji stratnej:

```
I=imread('cameraman.tif');
A=double(I)/255;       % konwersja klasy liczb
T=dctmtx(8);           % utworzenie macierzy transformacji
B=blkproc(A,[8 8],'P1*x*P2',T,T');
                       % obliczenie dwuwymiarowej DCT
mask=[1 1 1 1 0 0 0 0;
      1 1 1 0 0 0 0 0; % utworzenie maski do usuwania
      1 1 0 0 0 0 0 0; % nieznaczących współczynników DCT
      1 0 0 0 0 0 0 0; % z bloków obrazu
      0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0];
```

```
B2=blkproc(B,[8 8],'P1.*x',mask);
                    % usunięcie współczynników
I2=blkproc(B2,[8 8],'P1*x*P2','T',T);
                    % odwrotna DCT wykonywana
                    % na poszczególnych blokach
imshow(I);
figure, imshow(I2);  % wyświetlenie wyników
```

Aby przeprowadzić doświadczenia z różną liczbą usuwanych współczynników DCT i różnymi obrazami uruchom program `dctdemo`.

Przetwarzanie obrazów binarnych

Cel ćwiczenia

Celem ćwiczenia jest poznanie podstawowych funkcji biblioteki *Image Processing Toolbox (IPT)* pakietu MATALAB przeznaczonych do przetwarzania i analizy obrazów binarnych, m.in. funkcji implementujących metody *morfologii matematycznej* stosowanej do opisu i przetwarzania kształtu obiektów w obrazach. Przed przystąpieniem do ćwiczenia należy zapoznać się podstawami teoretycznymi elementarnych działań na zbiorach.

Przypomnijmy, że elementy obrazu binarnego przyjmują jedną z dwóch dyskretnych, umownie nadawanych wartości LOW=0 i HIGH=1. Tablice reprezentujące dyskretnego obrazy binarne w bibliotece IPT są klasy `double` lub `uint8`. W obrazach binarnych klasy `uint8` dopuszcza się również LOW=0 i HIGH=255.

Funkcje przetwarzania morfologicznego obrazów

Termin ***morfologia*** oznacza m.in. badanie kształtu i struktury. W przetwarzaniu obrazów wykorzystuje się morfologię matematyczną, dział teorii zbiorów, m.in. do analizy cech geometrycznych wyróżnionych obiektów obrazu. Metody morfologii matematycznej pozwalają rozpoznawać budowę obiektów a także przetwarzać ich kształt poprzez analizę badanego obrazu za pomocą specjalnych obiektów nazywanych elementami strukturującymi.

Poniżej podano definicje dwóch działań morfologicznych dylatacji i erozji, które są podstawą bardziej złożonych operacji morfologicznych. W definiowanych poniżej działaniach przyjmujemy, że zbiór *A* jest obiektem przetwarzanym a zbiór *B* jest elementem strukturującym, za pomocą którego wykonujemy poszczególne operacje morfologiczne. W obrazach binarnych przez obiekt rozumie się spójny zbiór punktów o wartościach HIGH.

Erozja

Niech *A* i *B* będą zbiorami w R^2. *Erozja A* przez *B* (gdzie *B* jest *elementem strukturującym*) jest działaniem postaci:

$$A \ominus B = \bigcap_{b \in B} (A + b)$$

Czyli A⊖B jest wspólną częścią wszystkich translacji *A* przez wszystkie elementy *B* o wektor *b* będący elementem zbioru *B*, dla każdego wektora $b \in B$ (zob. rys. 1).

Poniższa sekwencja poleceń ilustruje działanie erozji.

```
BW1 = imread('circbw.tif');    % wczytaj plik obrazowy
SE = ones(4,4);                % element strukturujący
BW2 = erode(BW1,SE);           % wykonaj erozję obrazu
imshow(BW1)                    % wyświetl obraz źródłowy
figure, imshow(BW2)            % wyświetl obraz wynikowy
```

Wykonaj podaną sekwencję poleceń i porównaj obrazy binarne BW1 i BW2.

Dylatacja

Niech A i B będą zbiorami w R^2. *Dylatację* A przez B (gdzie B jest tzw. *elementem strukturującym*) zdefiniuje się jako działanie:

$$A \oplus B = \bigcup_{b \in B} (A + b)$$

Innymi słowy, $A \oplus B$ jest wynikiem sumy zbiorów powstałych w wyniku translacji zbioru A o wektor b będący elementem zbioru B, dla każdego wektora $b \in B$ (zob. rys. 2).

Poniższa sekwencja poleceń ilustruje działanie dylatacji.

```
BW1 = imread('circles.tif');   % wczytaj plik obrazowy
SE  = ones(10,10);             % element strukturujący
BW2 = dilate(BW1,SE);          % wykonaj dylatację obrazu
imshow(BW1)                    % wyświetl obraz źródłowy
figure, imshow(BW2)            % wyświetl obraz wynikowy
```

Wykonaj podaną sekwencję poleceń i porównaj obrazy binarne BW1 i BW2.

Poniżej zdefiniowano operacje morfologiczne, które są wynikiem złożenia operacji dylatacji i erozji.

Otwieranie obiektu A elementem B definiuje działanie morfologiczne:

$$A \circ B = (A \ominus B) \oplus B,$$

czyli jest to ciąg operacji, w którym najpierw wykonuje się *erozję* obiektu A elementem strukturującym B a następnie uzyskany zbiór poddaje się *dylatacji* za pomocą tego samego elementu strukturującego (zob. rys. 3). W wyniku zastosowania operacji *otwierania* uzyskuje się: wygładzenie krawędzi i usuwanie przewężeń obiektu (np. otwieranie zamkniętych konturów).

Zamykanie obiektu A elementem B definiuje działanie morfologiczne:

$$A \bullet B = (A \oplus B) \ominus B,$$

czyli jest to ciąg operacji, w którym najpierw wykonuje się *dylatację* obiektu A elementem B a następnie uzyskany zbiór poddaje się *erozji* za pomocą tego samego elementu strukturującego. *Zamykanie* wygładza kontur obiektu oraz "zamyka" wąskie wgłębienia lub otwory w obiekcie(zob. rys. 3).

Do realizacji licznej grupy operacji morfologicznych na obrazach binarnych przewidziano w bibliotece IPT funkcję `'bwmorph'`.

Zapoznaj się ze składnią tej funkcji, która m.in. realizuje zdefiniowane wcześniej operacje morfologiczne dylatacji, erozji, otwierania oraz zamykania. Zwróć uwagę na inne liczne opcje działania tej funkcji (przekonaj się m.in. jak wyznaczyć tzw. szkielet obiektu). Zastosuj taki element morfologiczny, by za pomocą pojedynczej operacji otwierania pozostawić w obrazie płytki drukowanej `'circbw.tif'` pola stykowe (przy zachowaniu ich oryginalnych wymiarów) i usunąć wszystkie pozostałe ścieżki.

Inne funkcje przetwarzania obrazów binarnych biblioteki IPT

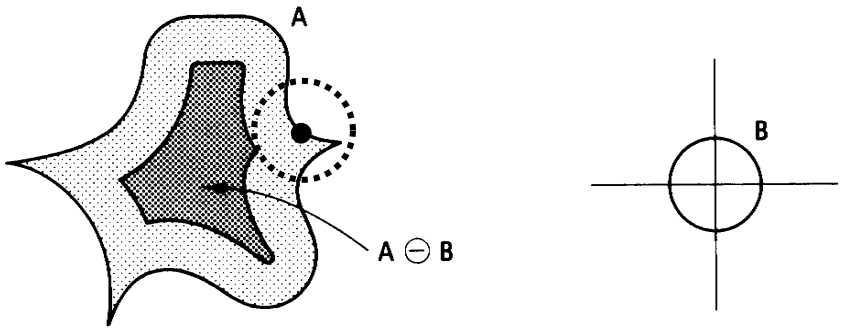
Biblioteka IPT jest również wyposażona w funkcje posiadające następujące możliwości przetwarzania i analizy obrazów binarnych:

- wyznaczanie brzegu obiektu w obrazie (‘bwperim’),
- wypełnianie zamkniętych konturów (‘bwfill’),
- etykietowanie obiektów (‘bwlabel’),
- interaktywne wydzielanie obiektów z obrazu (‘bwselect’),

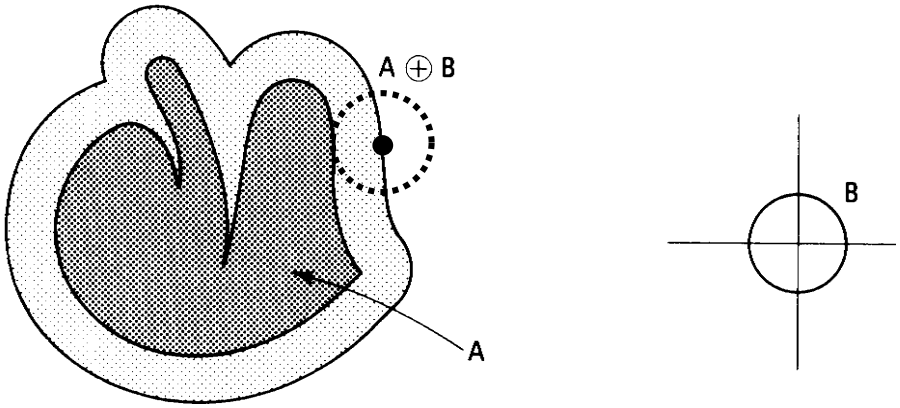
Poniżej podano przykładowe procedury ilustrujące działania wymienionych funkcji biblioteki IPT.

Zapoznaj się ze składnią funkcji zaznaczonych pogrubioną czcionką oraz oceń wyniki przetwarzania obrazów binarnych uzyskane za pomocą poniższych procedur.

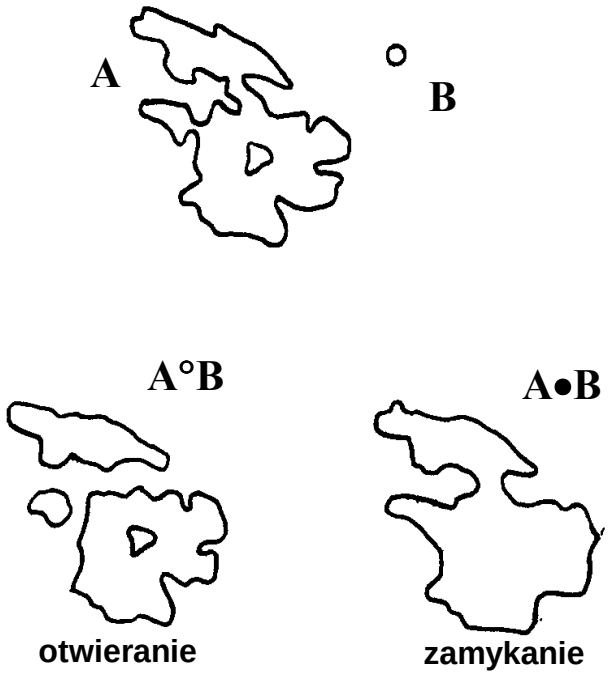
```
% BWPERIM
A=imread('circles.tif');imshow(A);
B=bwperim(A,8);
figure, imshow(B)

% BWLABEL
BW = [1 1 1 0 0 0 0 0
      1 1 1 0 1 1 0 0
      1 1 1 0 1 1 0 0
      1 1 1 0 0 0 1 0
      1 1 1 0 0 0 1 0
      1 1 1 0 0 0 1 0
      1 1 1 0 0 1 1 0 ];
L = bwlabel(BW,4)        % wyświetl zawartość tablicy L

% BWFILL
I = imread('blood1.tif');
BW3 = ~im2bw(I);
BW4 = bwfill(BW3,'holes');
imshow(BW3)
figure, imshow(BW4)

% BWSELECT
BW1 = imread('text.tif');
imshow(BW1);
pause;
c = [16 90 144];
r = [85 197 247];
BW2 = bwselect(BW1,c,r,4);
figure(2), imshow(BW2)
pause;
close figure(2);
bwselect   % to jest wersja interaktywna tej funkcji w
           % której wskazujesz myszką wybierane obiekty
           % obrazu + <ENTER>
```



Rys. 1. Erozja zbioru *A* elementem *B*



Rys. 2. Dylatacja zbioru *A* elementem *B*



Rys. 3. Ilustracja działania operacji morfologicznych otwierania i zamykania