Wydział Inżynierii Mechanicznej i Robotyki

Katedra Robotyki i Mechatroniki

Podstawy sztucznej inteligencji i uczenia głębokiego

Kurs dla kierunku Inżynieria Mechatroniczna

Instrukcja 6:

Wizja komputerowa: wprowadzenie i Metody wstępnego przetwarzania obrazu

Dowiesz się: Jak obrazy są reprezentowane w MATLAB-ie, czym są obrazy w skali szarości, RGB i obrazy binarne, jak załadować, przeglądać i zapisywać obrazy, jak manipulować macierzami obrazów i pojedynczymi pikselami, czym jest histogram obrazu i jak go rozumieć i nim manipulować, czym jest filtracja obrazu, czym są filtry liniowe i jak są realizowane matematycznie, czym są filtry nieliniowe, czym jest poprawa jakości obrazu – praktyczne zastosowania obu typów filtrów, czym są krawędzie w obrazie i jak są wykrywane, jakie są filtry morfologiczne stosowane do obrazów w skali szarości, czym jest progowanie jako najprostsza metoda segmentacji.

Materiały dodatkowe:

- Wykład do przedmiotu
- Zestaw obrazów testowych

Opiekun przedmiotu:
Krzysztof Holak, holak @agh.edu.pl

Autor instrukcji:
Krzysztof Holak, holak@agh.edu.pl

Materiały wprowadzające – praca z obrazami w MATLAB-ie

Instrukcje wykorzystują materiały dostępne w archiwum *VisionDatabase*– na stronie internetowej.

Przydatne polecenia:

`cls` – czyszczenie okna poleceń MATLAB-a z poleceń i wyników obliczeń
`clear all` – wyczyść wszystkie zmienne z przestrzeni roboczej MATLAB-a
`close all` – zamknij wszystkie otwarte figury
`ls` – wyświetl wszystkie pliki w katalogu roboczym

Aby załadować obraz do obszaru roboczego MATLAB-a, musimy określić zmienną, w której przechowujemy dane obrazu i użyć polecenia

```
A = imread('ścieżka');
```

Podajemy pełną ścieżkę do obrazu lub nazwę pliku obrazu z rozszerzeniem, jeśli obraz został skopiowany do katalogu, w którym pracujemy. Obraz zostanie zapisany w zmiennej macierzowej A. W zależności od załadowanego obrazu może to być reprezentacja obrazu w kolorach RGB lub w skali szarości. Aby sprawdzić liczbę kanałów kolorów i rozmiar obrazu (rozdzielczość obrazu), użyj polecenia

```
im_size = size(A);
```

Wektor im_size zawiera trzy liczby: M,N – liczbę pikseli w poziomie i w pionie (rozdzielczość obrazu) oraz C – liczbę kanałów kolorów.
Na przykład dla obrazu kolorowego możesz otrzymać na przykład: `im_size = [700 700 3];`

Dane obrazu są przechowywane w macierzy o rozmiarze MxNxC dla obrazów kolorowych lub MxN dla obrazów w skali szarości. Domyślnie trzy kanały kolorów reprezentują obrazy kolorowe w przestrzeni kolorów RGB, ale reprezentacja przestrzeni kolorów może zostać zmieniona na bardziej odpowiednią dla danego zadania.

Aby wyświetlić obraz, użyj polecenia

```
figure,
imshow(A);
```

Istnieją inne funkcje wyświetlania w MATLAB-ie, które pokazują macierze jako obrazy za pomocą pseudokolorowania. Są one przydatne do przeglądania wyników manipulacji obrazami i zostaną omówione w późniejszych ćwiczeniach laboratoryjnych.

W MATLAB-ie możesz pracować zarówno na obrazach kolorowych, jak i w skali szarości. Istnieją jednak funkcje, które działają tylko na obrazach w skali szarości. Dlatego wszystkie operacje na obrazach będą wykonywane na obrazach w skali szarości, chyba że zaznaczono inaczej. Aby zmienić przestrzeń kolorów z RGB na skalę szarości, użyj polecenia

```
A_gray = rgb2gray(A);
```

Pamiętaj, aby NIE używać tego polecenia na obrazie, który jest już w skali szarości. Otrzymasz błąd.

Możesz zapisać swoje obrazy po przetworzeniu jako pliki na dysku. Aby to zrobić, użyj polecenia (przykład)

```
imwrite(A_gray, 'ścieżka');
```

Ponownie możesz użyć całej ścieżki do nowego pliku lub tylko nazwy nowego pliku i rozszerzenia – plik obrazu zostanie zapisany w katalogu roboczym.
Pamiętaj, aby używać tego polecenia ostrożnie i nie nadpisywać istniejących plików obrazów!

Manipulacje macierzą obrazu

Możesz łatwo uzyskać dostęp do danych pikseli w zmiennej macierzy obrazu. Na przykład, jeśli chcesz wyciąć mniejszy obraz z oryginalnego obrazu, po prostu określ zakres wierszy i pikseli, jak poniżej (przykład)

```
B = A(20:100, 50:150);
```

W MATLAB-ie początek układu współrzędnych obrazu znajduje się w **lewym górnym rogu** obrazu. Dlatego **indeksy wierszy** rosną od góry do dołu obrazu, a **indeksy kolumn** rosną od lewej do prawej.

Teraz zobacz dwa małe przykłady przycinania obrazów:

1) Aby ręcznie przyciąć obraz, zaznacz lewy górny róg części obrazu, którą chcesz przyciąć, następnie wybierz zakres prostokąta przycinania, kliknij RPM na jego krawędzi i wybierz opcję przycinania obrazu z menu

   ```
   [A_small rectangle] = imcrop(A);
   ```

   Na wyjściu otrzymasz dwie zmienne: **`A_small`** – macierz z danymi pikselowymi przyciętego obrazu i **`rectangle`** – wektor zawierający lewy górny róg przyciętego prostokąta i jego rozmiar (szerokość i wysokość).

1) W przypadku automatycznego przycinania obrazu, jeśli znasz położenie i rozmiar części obrazu, która ma zostać przycięta (np. podczas przetwarzania zestawu obrazów), użyj tego samego polecenia, ale z **`rectangle`** jako drugim parametrem,

   ```
   rectangle = [x y width height];
   A_small = imcrop(A,rectangle);
   ```

Aby uzyskać dostęp do danych konkretnego piksela (współrzędnych x i y oraz wartości skali szarości lub RGB), użyj następującego polecenia

```
[x y value] = impixel(A);
```

Możesz wybrać tyle pikseli, ile chcesz. Po prostu kliknij LMB na pikselach, które wybierzesz

jeden po drugim. Aby zakończyć wybór pikseli, naciśnij Enter.

Zadanie 1.1: Proszę spróbuj manipulować macierzą obrazu w MATLAB-ie. Załaduj kolorowy obraz z zestawu obrazów dostarczonego przez wykładowcę, wyświetl go jako obraz kolorowy i obraz w skali szarości. Używając narzędzia impixel, w obu przypadkach odczytaj wartości pikseli z trzech różnych obszarów obrazu. Używając narzędzia imcrop, wytnij mały szczegół z dużego obrazu i zapisz go jako nową zmienną obrazu. Pokaż wektor z danymi na temat wyciętego obrazu – jego punkt początkowy oraz szerokość i długość.

Histogramy obrazu

W przypadku obrazów w skali szarości **histogram** jest wykresem przedstawiającym liczbę pikseli o danym poziomie intensywności (jasności) w obrazie. Należy tutaj wspomnieć, że jeśli obrazy są 8-bitowe, intensywność piksela może być przedstawiona za pomocą 256 wartości całkowitych (**uint8**) od 0 (czarny) do 255 (biały). Wszystkie liczby pomiędzy nimi oznaczają różne stopnie szarości.
Zmienna **double** w zakresie 0-1 będzie również interpretowana jako poziom jasności.
W przypadku obrazu kolorowego każdy kanał koloru ma 8 bitów, więc każdy piksel obrazu jest reprezentowany przez 24 bity. Pozwala to na ilościowe określenie ponad 16 milionów kolorów.
Histogram obrazu (i dowolnego innego typu danych) można również rozumieć jako funkcję masy prawdopodobieństwa, informującą o prawdopodobieństwie wystąpienia różnych wartości intensywności na obrazie.

W programie MATLAB histogram obrazu w skali szarości można wygenerować za pomocą poleceń

`imhist(A);` – do szybkiego obliczania i wizualizacji histogramu lub
`[value x] = imhist(A);` – jeśli chcesz zapisać dane histogramu do zmiennych. Możesz wizualizować histogram uzyskany w ten sposób, na przykład używając polecenia **`bar(x, value)`**.

Teraz zobacz proste polecenie, które ładuje obraz, oblicza i wyświetla jego histogram

```
clc
clear all
close all

imRGB = imread('Mountains.jpg');

figure(1)
subplot(1,2,1);
imshow(imRGB);

imGray = rgb2gray(imRGB);

figure(1)
subplot(1,2,2);
imshow(imGray);

imDataRGB = size(imRGB);
imDataGray = size(imGray);

[value coordinate] = imhist(imGray);

figure(2)
subplot(1,2,1);
imshow(imGray);
subplot(1,2,2);
bar(coordinate, value);
```

Teraz wykonaj zadania praktyczne, aby lepiej zrozumieć ideę histogramu obrazu.

Wyrównanie histogramu

Jedną z podstawowych operacji na histogramach jest **wyrównywanie histogramu** . Stosuje się ją zazwyczaj, gdy obraz ma niski kontrast. Wprowadzona zmiana w rozkładzie poziomów jasności na obrazie „spłaszcza" histogram, co w praktyce zwiększa jego kontrast wizualny. Taka operacja może na przykład ujawnić pewne struktury, które wcześniej były niewidoczne dla ludzkiego oka z powodu niskiego kontrastu. Aby wyrównać histogram obrazu, należy użyć funkcji **`histeq(A)`** .

```
[A_enh T] = histeq(A);
```

Argumentem funkcji jest obraz w skali szarości, ale tę operację można również zastosować do obrazów RGB. Wyjściem jest ulepszony obraz i użyta transformacja wartości intensywności.

Zadanie 1.2: Oblicz i narysuj histogramy dla obrazów (w skali szarości) wybranych z zestawu obrazów dostarczonego przez instruktora. Narysuj obraz i jego histogram na tej samej figurze w programie Matlab. Przeprowadź wyrównanie histogramu. Narysuj wynikowy obraz i jego histogram w drugiej figurze.

Zadanie 1.3: Utwórz dwa obrazy szumu losowego za pomocą generatorów liczb losowych MATLAB-a. W pierwszym przypadku szum powinien być jednorodny (z płaskim histogramem), a w drugim przypadku szum powinien być normalny (jego histogram jest krzywą dzwonową Gaussa, wyśrodkowaną na średniej intensywności szarości). Napisz odpowiedni kod za pomocą funkcji rand() i randn().

Pamiętaj, jaki jest zakres wartości, jakie może przyjąć piksel w skali szarości (0-255 dla typu uint8 lub 0-1 dla double)! Jeśli generator liczb losowych zwraca liczby spoza tego

zakresu, musisz przeskalować wynik.

Użyj większych obrazów, jeśli chcesz, aby uzyskany histogram przypominał teoretyczny histogram dla rozkładu równomiernego i normalnego.

Zastosowanie histogramów – proste progowanie

Segmentacja obrazu polega na podziale obrazu na obiekty i tło. Jest to bardzo szerokie zagadnienie, obejmujące zarówno najprostsze algorytmy, w których klasyfikacja każdego piksela do danego obiektu lub tła jest określana przez jego intensywność, jak i bardzo złożone metody segmentacji semantycznej wykorzystujące głębokie sieci neuronowe.

Najprostszym algorytmem, ale bardzo ważnym w praktyce, jest **progowanie (binaryzacja)** . Polega ono na przekształceniu obrazu w skali szarości w obraz czarno-biały. Jest wykonywane na podstawie **progu binaryzacji**. Na przykład pikselom o wartości mniejszej od progu nadawana jest wartość 0, a pikselom o wartości większej od progu nadawana jest wartość 1. Binaryzacja jest często pierwszym krokiem w wielu algorytmach analizy obrazu. Pozwala podzielić scenę na obiekty i tło, jeśli jasność ich pikseli kontrastuje ze sobą i może być oddzielona jednym lub kilkoma progami. Wartość progu można znaleźć ręcznie za pomocą inspekcji histogramu. Na przykład, jeśli histogram obrazu jest bimodalny – z dwoma szczytami odpowiadającymi odpowiednio obiektom i tłu, środkowy poziom intensywności między nimi może zostać wybrany jako próg binaryzacji. Można również zastosować automatyczne progowanie, na przykład za pomocą **metody Otsu** .

Zobacz poniższy kod, który wykonuje prostą binaryzację.

```
clc
clear all
close all

imRGB = imread('Mountains.jpg');

figure(1)
subplot(1,3,1);
imshow(imRGB);

imGray = rgb2gray(imRGB);

figure(1)
subplot(1,3,2);
imshow(imGray);


threshold = % fill in your threshold value!
% write here the value of the threshold you found by insepcting the histogram of your
image or apply automatic thresholding threshold = graythresh(imGray); in MATLAB, the
function im2bw needs threshold in double 0, it is a number between 0 and 1!

imBW = im2bw(imGray, threshold);
% you can use also logical statements, for example imBW = imGray < threshold


figure(1)
subplot(1,3,3);
imshow(imBW);
```

Jeśli używasz funkcji **`im2bw()`**, pamiętaj, że wymaga ona, aby wartość progowa była liczbą z przedziału od 0 do 1. Jeśli odczytujesz wartość progową z histogramu utworzonego dla obrazu zawierającego wartości całkowite, musisz ją najpierw znormalizować do przedziału

od 0 do 1.

Zadanie 1.4 : Używając powyższego kodu, wykonaj binaryzację obrazu z Image Set zgodnie z prośbą instruktora. W powstałym obrazie jeden obiekt wybrany przez instruktora musi być biały, a pozostałe piksele muszą być ustawione na zero.

Podstawy filtrowania obrazu

Najpierw krótkie wprowadzenie teoretyczne. Obraz można rozumieć jako sygnał 2D, w którym zmiennymi niezależnymi są współrzędne położenia (x, y) piksela w układzie współrzędnych obrazu, a zmienną zależną jest jasność piksela (w przypadku obrazów w skali szarości) lub wektor 3 wartości opisujących kolor piksela (w przypadku obrazów RGB).

Filtrowanie jest podstawową operacją kontekstową wykonywaną na obrazach. Filtry liniowe są implementowane jako splot obrazu z maskami o różnych współczynnikach liczbowych w zależności od ich funkcji. Filtry liniowe dzielą się na filtry dolnoprzepustowe i górnoprzepustowe. Podobnie jak w przypadku sygnałów jednowymiarowych, filtry dolnoprzepustowe usuwają częstotliwości przestrzenne wyższe niż dana częstotliwość odcięcia z obrazu, a filtry górnoprzepustowe usuwają niskie częstotliwości, zachowując jednocześnie wysokie częstotliwości w obrazie.

W poniższym przykładzie dowiesz się, czym jest częstotliwość przestrzenna i jakie niskie i wysokie częstotliwości przestrzenne występują w obrazie. Za jakie elementy obrazu odpowiadają te częstotliwości i co się stanie, jeśli zostaną usunięte (lub zredukowane) z obrazu przez proces filtrowania?

Istnieje funkcja **`conv2()`**, która oblicza splot sygnałów dwuwymiarowych. Ta funkcja przyjmuje obraz i przygotowaną przez użytkownika maskę filtra jako argumenty. Wynikiem jest nowy obraz po filtrowaniu. Wynik jest przechowywany w macierzy liczb double, więc możesz przekonwertować liczby na uint8 przed wyświetleniem. W tym ćwiczeniu zobaczysz przykłady filtra dolnoprzepustowego i górnoprzepustowego.

Jako przykład filtra dolnoprzepustowego bierzemy najprostszy **prostokątny filtr uśredniający** (box). Jego maska dla rozmiaru 3x3 to

```
mask1 = (1/9)*[1 1 1; 1 1 1; 1 1 1];
```

Filtrowanie przy użyciu wcześniej wspomnianej funkcji wykonuje się w następujący sposób

```
imFilt1 = conv2(image, mask1);
```

Jako przykład filtra górnoprzepustowego przyjrzyjmy się **operatorowi Sobela** (nazwa pochodzi od imienia Irwina Sobela, urodzonego w 1940 r.; filtr ten opracowano w 1968 r. w Stanford AI Project w ramach pracy doktorskiej).

```
mask2 = [1 0 -1; 2 0 -2; 1 0 -1];
```

Prosty kod do testowania tych filtrów

```
clc
clear all
close all

imRGB = imread('Mountains.jpg');
imGray = rgb2gray(imRGB);

figure(1)
subplot(1,3,1);
imshow(imGray);

mask1 = (1/9)*[1 1 1; 1 1 1; 1 1 1]; % LP box filter
mask2 = [1 0 -1; 2 0 -2; 1 0 -1];    % HP Sobel mask (directional)

im_filt1 = conv2(imGray), mask1);
im_filt2 = conv2(imGray), mask2);

figure(1)
subplot(1,2,3);
imshow(uint8(im_filt1));

figure(1)
subplot(1,3,3);
imshow (uint8(im_filt2));
```

Wykonajmy proste ćwiczenia wykorzystując podany kod.

Zadanie 1.5: Uruchom kod podany w instrukcji laboratoryjnej. Wybierz obraz z zestawu obrazów podanego przez wykładowcę, wyświetl wyniki filtracji przy użyciu obu filtrów liniowych. Zmień rozmiar maski 1 z 3x3 na 5x5 i 7x7. Użyj sumy jedynek w masce jako stałej normalizacyjnej (1/9 w powyższym przykładzie). Wyświetl wyniki. Transponuj maskę 2 i wyświetl wyniki filtracji.

Filtracja dolnoprzepustowa – filtry liniowe

Ze względu na duże praktyczne znaczenie filtracji obrazu, MATLAB ma specjalistyczną funkcję **`imfilter()`** dedykowaną tej operacji. Jeśli chcesz użyć standardowych filtrów, ich maski można łatwo uzyskać za pomocą funkcji pomocniczej **`fspecial()`**. Podaj nazwę filtra i podstawowe parametry związane z każdym z nich.

Na przykład, aby zastosować nasz filtr typu box z poprzedniego ćwiczenia, użyj

```
h = fspecial('average',[3 3]);
im_filt = imfilter(imGray,h,'replicate');
```

Wykonajmy nasze poprzednie ćwiczenie używając tego kodu

```
clc
clear all
close all

imRGB = imread('Mountains.jpg');
imGray = rgb2gray(imRGB);

figure(1)
subplot(1,3,1);
imshow(imGray);

mask1 = fspecial('average',3); % LP box filter
mask2 = fspecial('sobel');     % HP Sobel mask (directional)

im_filt1 = imfilter(imGray,mask1);
```

```
im_filt2 = imfilter(imGray,mask2');

figure(1)
subplot(1,2,3);
imshow(im_filt1);

figure(1)
subplot(1,3,3);
imshow (im_filt2);
```

Najpierw zajmiemy się filtrami dolnoprzepustowymi. Filtracja dolnoprzepustowa jest wykonywana przez następujące typy filtrów dostępnych w funkcji fspecial :

1) Prostokątny filtr uśredniający o rozmiarze okna podanym przez zmienną hsize (nieparzysta liczba całkowita)
   `h = fspecial('average',hsize);`

2) Filtr uśredniający kołowy o rozmiarze podanym przez zmienną radius
   `h = fspecial('disk',radius);`

3) Filtr Gaussa wymagający rozmiaru maski i odchylenia standardowego krzywej dzwonowej, która jest aproksymowana przez współczynniki maski
   `h = fspecial('gaussian',hsize,sigma)`

Teraz wykonaj następujące ćwiczenie

Zadanie 1.6: Proszę wykonać filtrację dolnoprzepustowe obrazu z zestawu obrazów. Wyświetl obrazy otrzymany w wyniku filtracji za pomocą filtrów average i disk. Wypróbuj trzy różne rozmiary masek.
Po skorzystaniu z pomocy programu Matlab wybierz dwie różne wartości parametru hsize i sigma dla filtru Gaussa i wyświetl wyniki filtracji obrazu z tymi ustawieniami filtra.

Filtrowanie nieliniowe

Najczęściej stosowanym filtrem nieliniowym w przetwarzaniu obrazu jest **filtr medianowy**. Ten filtr określa medianę dla każdego piksela, tj. wartość centralną jasności pikseli w oknie pod maską filtra, której środek znajduje się w tym pikselu. W MATLAB-ie filtrowanie medianowe można wykonać za pomocą funkcji **`medfilt2()`**.

Przykład użycia filtra medianowego:

```
im_noisy = imnoise(imGray,'salt & pepper',0.02);
im_filtered = medfilt2(im_noisy);
```

W przykładzie wykorzystano funkcję **`imnoise()`** programu MATLAB, która dodaje do obrazu szum o podanym rozkładzie statystycznym (w przykładzie „sól i pieprz” to szum impulsowy) o zadanym natężeniu (w przykładzie 2% zmodyfikowanych pikseli).

Zadanie 1.7 : Popraw jakość obrazu, usuwając szumy lub rysy. Zastosuj odpowiednie filtry, aby usunąć szumy obecne na obrazie dostarczonym przez instruktora.

Filtr górnoprzepustowy i wykrywanie krawędzi

Istnieje wiele metod wykrywania krawędzi, z których zajęcia laboratoryjne będą omawiać

tylko te oparte na liniowym filtrowaniu górnoprzepustowym. Filtry te numerycznie obliczają pierwszą lub drugą pochodną obrazu. Aby określić maski tego typu filtrów, możesz użyć już znanych funkcji **`imfilter()`**i **`fspecial()`**. Druga z nich pozwala na wygenerowanie następujących filtrów:

Operatory, które numerycznie obliczają pierwszą pochodną obrazu. Są kierunkowe, tzn. dają maksymalną odpowiedź na pionowe lub poziome krawędzie. Możesz zmienić ich kierunkowość , wykonując transpozycję maski

1) **Operator Sobela**
   `h = fspecial('sobel');`

2) **Operator Prewitt** (nazwa pochodzi od Judith Prewitt, praca z 1970 r.)
   `h = fspecial('prewitt');`

Operatory, które numerycznie obliczają sumę drugich pochodnych cząstkowych względem zmiennych x i y. W MATLAB-ie są dostępne dwa - **operator Laplace'a** i **Laplace of Gaussian** (**LoG**). Są to operatory niekierunkowe (w pewnym zakresie, bo działają na przestrzeni punktów dyskretnych w regularnej siatce prostokątnej).

3) **Operator Laplace'a** - zmienna alfa określa kształt Laplacianu (zobacz pomoc MATLAB-a, dla alfa = 0 otrzymujemy standardowy Laplacian)
   `h = fspecial('laplacian',alpha);`

4) **Operator LoG** – przed obliczeniem pochodnych obraz jest rozmywany metodą Gaussa, aby zmniejszyć wpływ szumu na wykrywanie krawędzi. Znaczenie parametrów jest takie samo jak w przypadku filtra Gaussa.
   `h = fspecial('log', hsize,sigma);`

Alternatywą dla tego podejścia jest funkcja **`edge()`**, która jest dedykowana wyłącznie do wykrywania krawędzi w obrazach. Funkcje te wykorzystują operatory opisane powyżej. Krawędzie znajduje dla pikseli, dla których pierwsza pochodna ma maksimum lub w punktach przejścia przez zero drugiej pochodnej.
Na przykład, aby zastosować filtr Sobela za pomocą tej funkcji, użyj następującego kodu

```
im_edge = edge(imGray,'sobel',thresh,direction);
```

Zmienna thresh określa odpowiedź filtra, powyżej której piksel jest uważany za należący do krawędzi, a direction określa kierunek, w którym obliczana jest pierwsza pochodna obrazu ('horizontal' lub 'vertical'). Ta funkcja pozwala na używanie operatorów Sobela, Prewitta i logarytmicznych, jak poprzednio.
Ponadto oferuje nowe typy detektorów krawędzi:

5) **Roberts' Cross** – nazwa pochodzi od Lawrence'a Robertsa (1937-2018), praca z 1963 r., L. Roberts był jednym z twórców ARPANET i jest nazywany jednym z ojców założycieli Internetu. Jego filtr jest najprostszym detektorem krawędzi.

6) **Detektor krawędzi Canny'ego** – jeden z najsłynniejszych algorytmów wizyjnych stosowanych w praktyce. Posiada dwa progi, których prawidłowe ustawienie pozwala na określenie nieprzerwanych krawędzi przy jednoczesnej redukcji szumów. John Canny (ur. 1958) jest profesorem na University of Berkeley w

Kalifornii. Praca na temat detektora pochodzi z 1986 r.

Teraz wykonuje szereg zadań dotyczących wykrywania krawędzi

Zadanie 1.8: Zastosuj wykrywanie krawędzi do obrazu dostarczonego przez instruktora. Zastosuj filtry górnoprzepustowe: Sobela, Prewitta i Laplaciana za pomocą funkcji imfilter() i wyświetl wyniki, zastosuj funkcję edge() dla tego samego typu filtrów co poprzednio i wyświetl wynik.

Zadanie 1.9: Wykrywanie krawędzi za pomocą metody filtrowania Canny'ego. Wybierz parametry filtra, aby uzyskać krawędzie obiektów na obrazie dostarczonym przez instruktora.

Filtracja morfologiczna

W tej części nauczysz się, jak używać filtrów morfologicznych na obrazach w skali szarości. W następnym laboratorium te filtry zostaną omówione znacznie bardziej szczegółowo w przypadku obrazów binarnych. Mówiąc ogólnie, w przypadku obrazów w skali szarości filtry te działają jak filtry max lub min. Zastępują wartość piksela wartością, która jest najwyższa/najniższa w masce wyśrodkowanej na tym pikselu. Użytkownik może dowolnie wybierać kształt maski filtra.
Aby użyć tego filtra, najpierw musisz utworzyć element strukturalny, jak w przykładzie

```
se1 = strel('type',size);
```

Wybierz kształt maski zgodnie z zadaniem, które zamierzasz wykonać. Następnie możesz zastosować następujące cztery operacje do obrazu wejściowego - erozję, dylatację, otwieranie i zamykanie.

```
bw_erode = imerode(im_bw,se1);
bw_dilate = imdilate(im_bw,se1);
bw_open = imopen(im_bw,se1);
bw_close = imclose(im_bw,se1);
```

Teraz wykonaj ostatni zestaw zadań z instrukcji.

Zadanie 1.10: Zastosuj cztery podstawowe operacje na obrazie z zestawu obrazów. Narysuj wyniki czterech podstawowych filtrowań morfologicznych na jednym rysunku w programie Matlab. Pokaż wyniki dla dwóch różnych kształtów elementów strukturalnych i 2 różnych rozmiarów maski.

Zadanie 1.11: Popraw jakość obrazów poprzez usunięcie szumu lub zarysowań, używając wyłącznie operacji morfologicznych. Zastosuj odpowiednie filtry, aby usunąć szum obecny na dostarczonych obrazach.

Dodatkowe zadania

Zadanie 1.12: Proszę wykonać ćwiczenie 1.1, ale użyć obrazów w kolorze RGB zamiast obrazów w skali szarości. Pamiętaj, że musisz przetworzyć każdy kanał koloru osobno, a następnie samodzielnie utworzyć histogramy RGB. Spróbuj przekonwertować przestrzeń kolorów RGB na HSV. Używając różnych obrazów w kolorze, spróbuj zrozumieć, co reprezentują kanały HSV. Utwórz histogramy dla obrazów w tej reprezentacji i porównaj je z histogramami utworzonymi przy użyciu obrazów RGB.

Zadanie 1.13: Jeśli ukończyłeś ćwiczenie 1.3, powtórz ćwiczenie 1.4 dla obrazów w kolorze RGB. Omów różnice między obrazami przed i po zauważonym ulepszeniu zarówno na obrazach, jak i na ich histogramach RGB.

Zadanie 1.14 : Wygeneruj serię zaszumionych obrazów z szumem impulsowym i gaussowskim poprzez zastosowanie funkcji imnoise() poprzez sukcesywne zwiększanie parametrów szumu. Następnie wypróbuj filtry, których się nauczyłeś, aby usunąć szum tak skutecznie, jak to możliwe, bez znaczącej zmiany zawartości obrazu. Wypróbuj różne rozmiary masek filtrów i różne wartości innych parametrów.
Możesz pokazać poziom szumu obecnego w poprzez przycięcie części obrazu o stałej intensywności, a następnie wygenerowanie dla niej histogramu. Histogram obrazu bez szumu powinien być zaciskany wokół podobnych wartości intensywności. Jako numeryczną miarę szumu możesz użyć odchylenia standardowego części obrazu o stałej intensywności.

Zadanie 1.15: Wygeneruj serię zaszumionych obrazów z szumem impulsowym i gaussowskim poprzez zastosowanie funkcji imnoise() poprzez sukcesywne zwiększanie parametrów szumu. Następnie spróbuj użyć filtrów morfologicznych, aby usunąć szum tak skutecznie, jak to możliwe, bez znaczącej zmiany zawartości obrazu. Wypróbuj różne rozmiary i kształty masek filtrów.
Możesz pokazać poziom szumu obecnego w poprzez przycięcie części obrazu o stałej intensywności, a następnie wygenerowanie dla niej histogramu. Histogram obrazu bez szumu powinien być zaciskany wokół podobnych wartości intensywności. Jako numeryczną miarę szumu możesz użyć odchylenia standardowego części obrazu o stałej intensywności.