Laboratorium 1

Odczytywanie i zapisywanie obrazów rastrowych do plików, operacje punktowe na tablicach obrazów

Konfiguracja systemu WinPython 3.6/ 3.7

Otworzyć konsolę Python'a „WinPython Command Prompt" i sprawdzić obecność modułów wykorzystywanych do obróbki obrazów. W tym celu wylistować wszystkie zainstalowane pakiety poleceniem **pip3 freeze**. Na liście odszukać pakiety o nazwach: **scipy, numpy, matplotlib, scikit-image, opencv-python, pydicom**. Pakiety nieobecne na liście doinstalować kolejno poleceniem **pip3 install [nazwa_modułu]**. Aktualizacja modułów wymaga wykonania polecenia **pip3 install [nazwa modułu] --upgrade**.

Pakiet scipy tworzy podstawowe środowisko dla operacji matematycznych, obliczeń naukowych I inżynierskich realizowanych przez pakiety wymienione poniżej.

Te dodatkowe pakiety najlepiej importować w treści skryptu poleceniami:

import numpy as np

podstawowy pakiet obliczeń numerycznych z wykorzystaniem tablic wielowymiarowych ndarray obejmujący operacje na tych tablicach

import skimage as skimg
from skimage import io
from skimage import color

skimage (scikit-image) to pakiet podstawowych procedur przetwarzania obrazów do swobodnego wykorzystania, bez ograniczeń licencyjnych.

skimage.io – moduł procedur odczytu plików obrazowych do tablic w pamięci operacyjnej, wyświetlania obrazów w oknie i zapisu tablic obrazów do pliku binarnego w formatach specyficznych dla danego typu obrazu.

skimage.color – moduł procedur różnych transformacji wielowymiarowych tablic obrazów ndarray przestrzeni koloru

import pydicom as dicom

pakiet procedur do obróbki obrazów zapisanych w formacie plików DICOM wykorzystywanych w medycynie i biologii.

import cv2

pakiet opencv-python zawierający niezależną bibliotekę obróbki obrazów OpenCV stworzoną oryginalnie w języku C++. W bieżącym ćwiczeniu można ją zastosować przy zapisie i odczycie wieli

obrazów z pliku oraz do wyświetlania obrazów dwuwymiarowych i wpisywania do nich różnych markerów

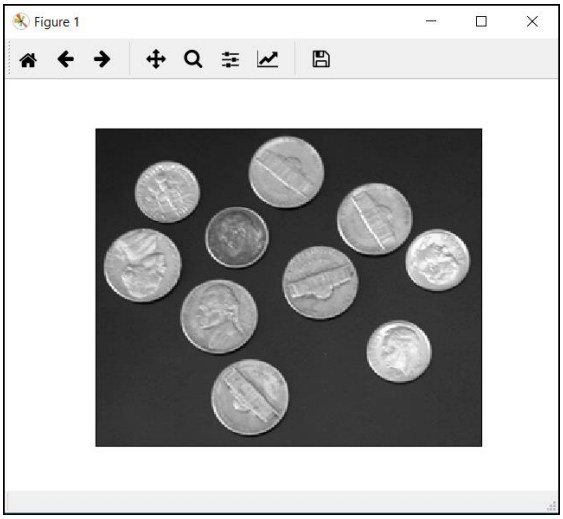
import matplotlib.pyplot as plt
from matplotlib.pyplot import cm
import matplotlib.image as mpimg
import matplotlib.patches as patches

moduły pakietu matplotlib umożliwiające odczyt i zapis obrazów oraz tworzenie napisów, wykresów i wyświetlanie obrazów w oknie.

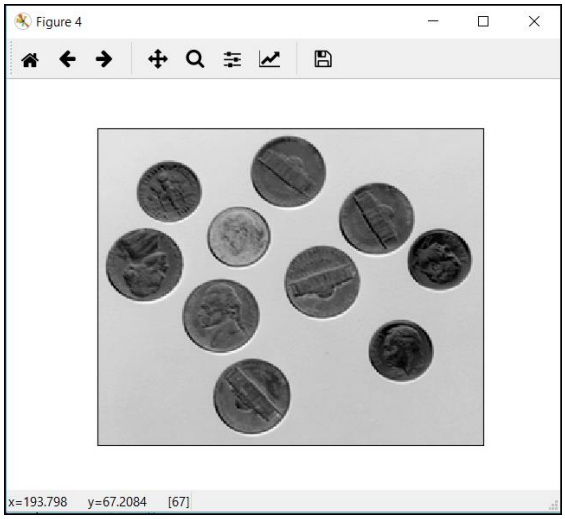
Przepisać do folderu lokalnego ‘images’ obrazy testowe pobrane z zasobu katalogowego o tej samej nazwie w bazie WIKAMP.
Uruchomić środowisko Spyder3 z konsolą IPython do wykonywania podanych poniżej zadań w formie różnych funkcji skryptu ‘input_output.py’.

Zadanie 1

Odczytać kolejno wszystkie pliki obrazowe (tif, jpg, png, bmp) z podkatalogu ‘input1’ przy pomocy funkcji matplotlib.image.imread i wyświetlić ich treści w wyskakującym oknie przy pomocy funkcji matplotlib.pyplot.imshow jako obrazy w odcieniach szarości lub obrazy kolorowe (true color). Następnie odwrócić poziomy jasności (składowych koloru) w tablicach obrazów. Odwrócone obrazy wyświetlić w nowym oknie i następnie zapisać w podkatalogu ‘output’ jako pliki obrazów tego samego typu (matplotlib.image.imsave), z nazwą oryginalnego pliku uzupełnioną cyfrą 2 (os.path.split, os.path.basename, os.path.splitext, lub filename.split). Usunąć opisy osi funkcjami matplotlib.pyplot.axis('off') lub matplotlib.pyplot..xticks([ ]), matplotlib.pyplot..yticks([ ]) .

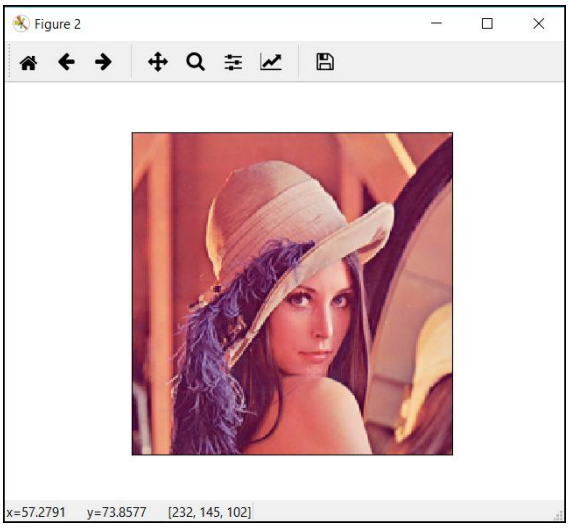


a)

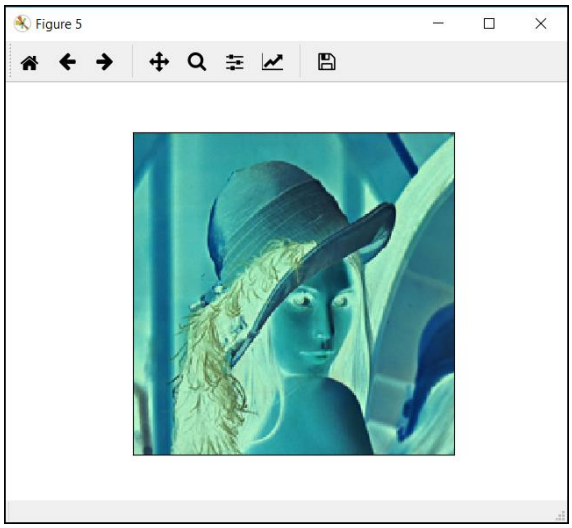


b)

Rys. 1. a) Obraz z poziomami jasności odczytany w osobnym oknie, b) obraz z odwróconymi poziomami jasności



a)

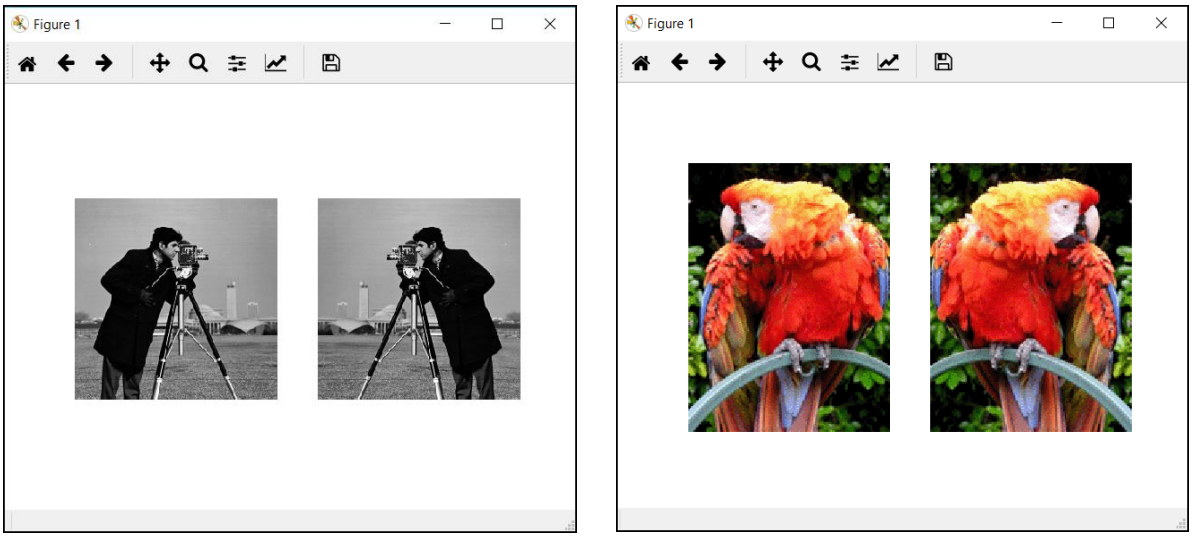


b)

Rys. 2. a) Obraz kolorowy RGB odczytany w osobnym oknie, b) obraz RGB z odwróconymi poziomami składowych koloru

Zadanie 2

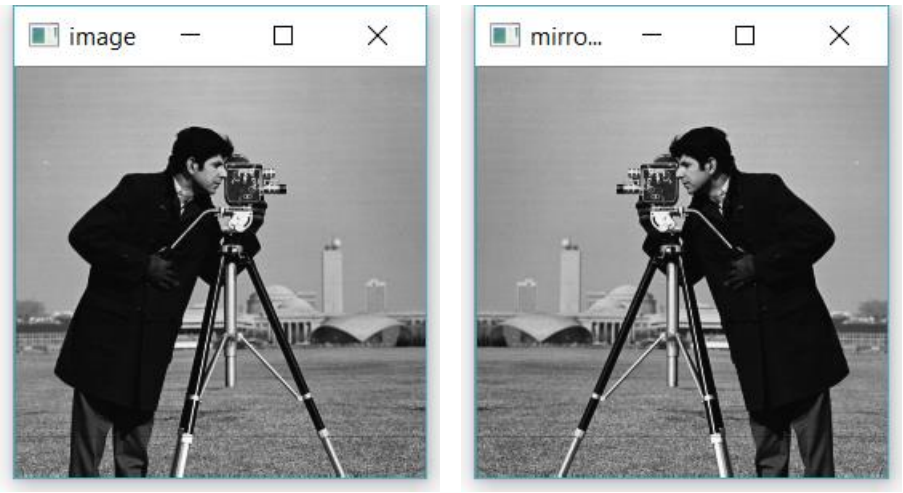
Odczytać kolejno wszystkie pliki obrazowe (tif, jpg, png, bmp, gif) z podkatalogu ‘input1’ przy pomocy funkcji skimage.io.imread i wykonać ich kopie, na których należy odwrócić oryginalną treść obrazów na zasadzie zwierciadła. Wyświetlić obok siebie, w jednym wierszu obraz oryginalny i odwrócony, w jednym oknie wyskakującym przy pomocy funkcji matplotlib.pyplot. Na obrazach usunąć skale na osiach. Zmodyfikowane kopie obrazów zapisać w podkatalogu ‘output’ jako pliki obrazów tego samego typu (przy pomocy skimage.io.imsave), z nazwą oryginalnego pliku uzupełnioną cyfrą 3.



Rys. 3. Obrazy wraz ze swoim odbiciami zwierciadlanymi wyświetlone w jednym wierszu tego samego okna, a) obraz z poziomami jasności, b) obraz kolorowy RGB

Zadanie 3

Powtórzyć zadanie 2 korzystając z funkcji odczytu pliku cv2.imread w pakiecie opencv, także z funkcji wyświetlania cv2.imshow i zapisu pliku cv2.imwrite. Zapisać do dysku również obrazy luminancji o głębokości 8 bit (funkcje cv2. cvtColor lub skimage.color.rgb2gray)



Rys. 4. Obraz z poziomami jasności i jego odbicie zwierciadlane wyświetlone w osobnych oknach przy użyciu funkcji z pakietu opencv



a) b) c)

Rys. 5. Obraz kolorowy RGB a) i jego odbicie zwierciadlane b) oraz odbicie zwierciadlane luminancji obrazu c) wyświetlone w osobnych oknach przy użyciu funkcji z pakietu opencv

Zadanie 4

Napisać funkcję contrast_stretch do rozciągania kontrastu w obrazie 2D z wieloma poziomami jasności, wyświetlającą obraz po załadowaniu z pliku i obraz wynikowy z rozciągniętym kontrastem w zakresie percentyli od 2% do 98% podanym jako parametr funkcji (numpy.percentile). Wyświetlić także histogram obrazu przed i po rozciągnięciu dla różnej ilości słupków (matplotlib.pyplot.hist). Zastosować funkcję wbudowaną (skimage.exposure.rescale_intensity).

Zadanie 5

Napisać funkcję contrast_stretch_3D do rozciągania kontrastu w obrazie 3D, na jednym z obrazów DICOM 3D w katalogu’\images\input2’. wyświetlającą obraz po załadowaniu i obraz wynikowy z rozciągniętym kontrastem w zakresie percentyli od 2% do 98% podanym jako parametr funkcji (numpy.percentile). Wyświetlić także histogram obrazu przed i po rozciągnięciu dla różnej ilości słupków (matplotlib.pyplot.hist).

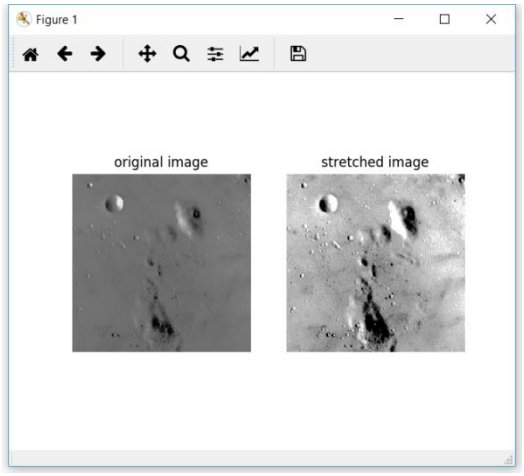


Figure 1
original image
stretched image

a)

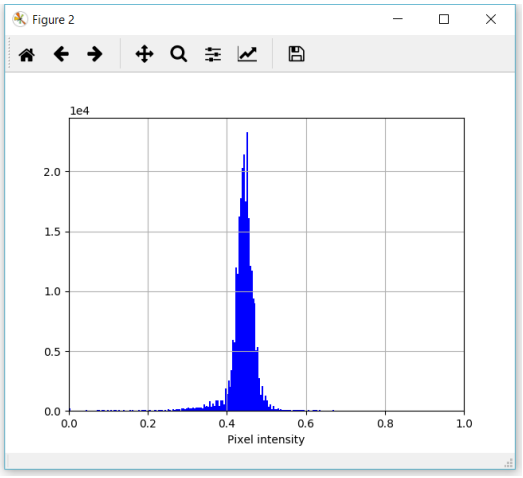


Figure 2
1e4
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Pixel intensity

b)

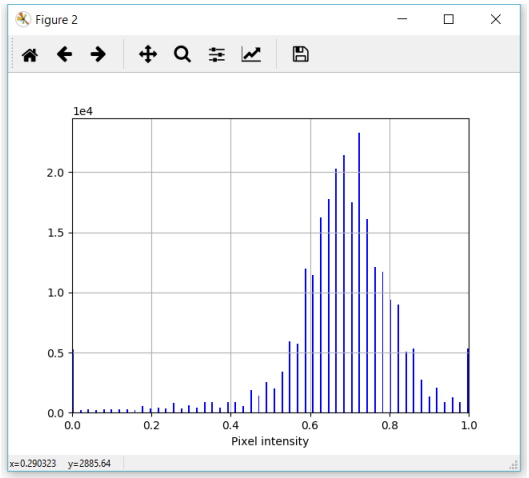


Figure 2
1e4
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Pixel intensity
x=0.290323 y=2885.64

c)

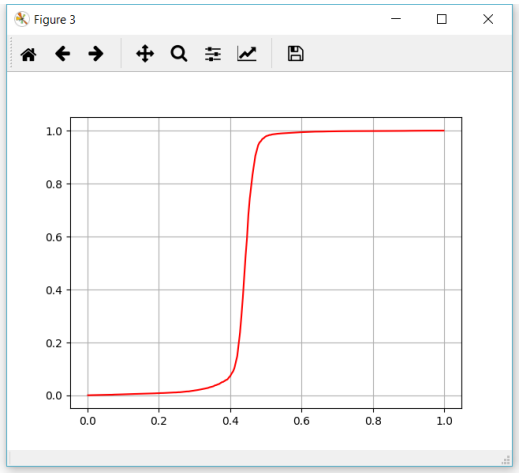


Figure 3
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

d)

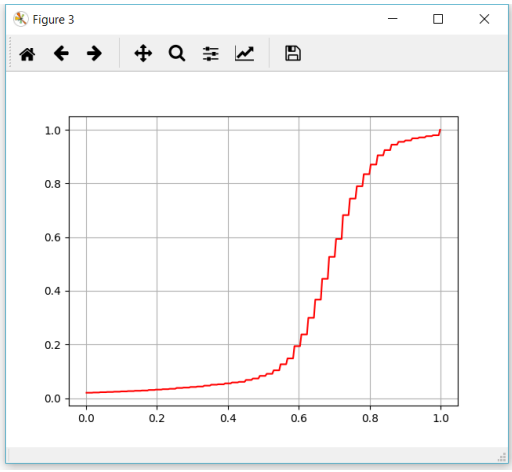


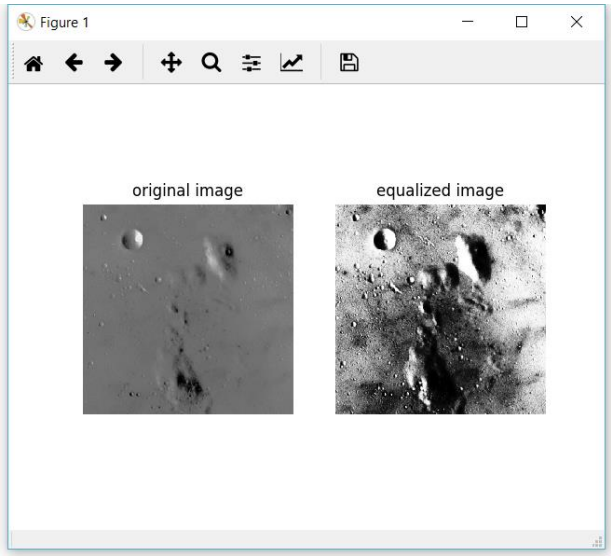
Figure 3
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

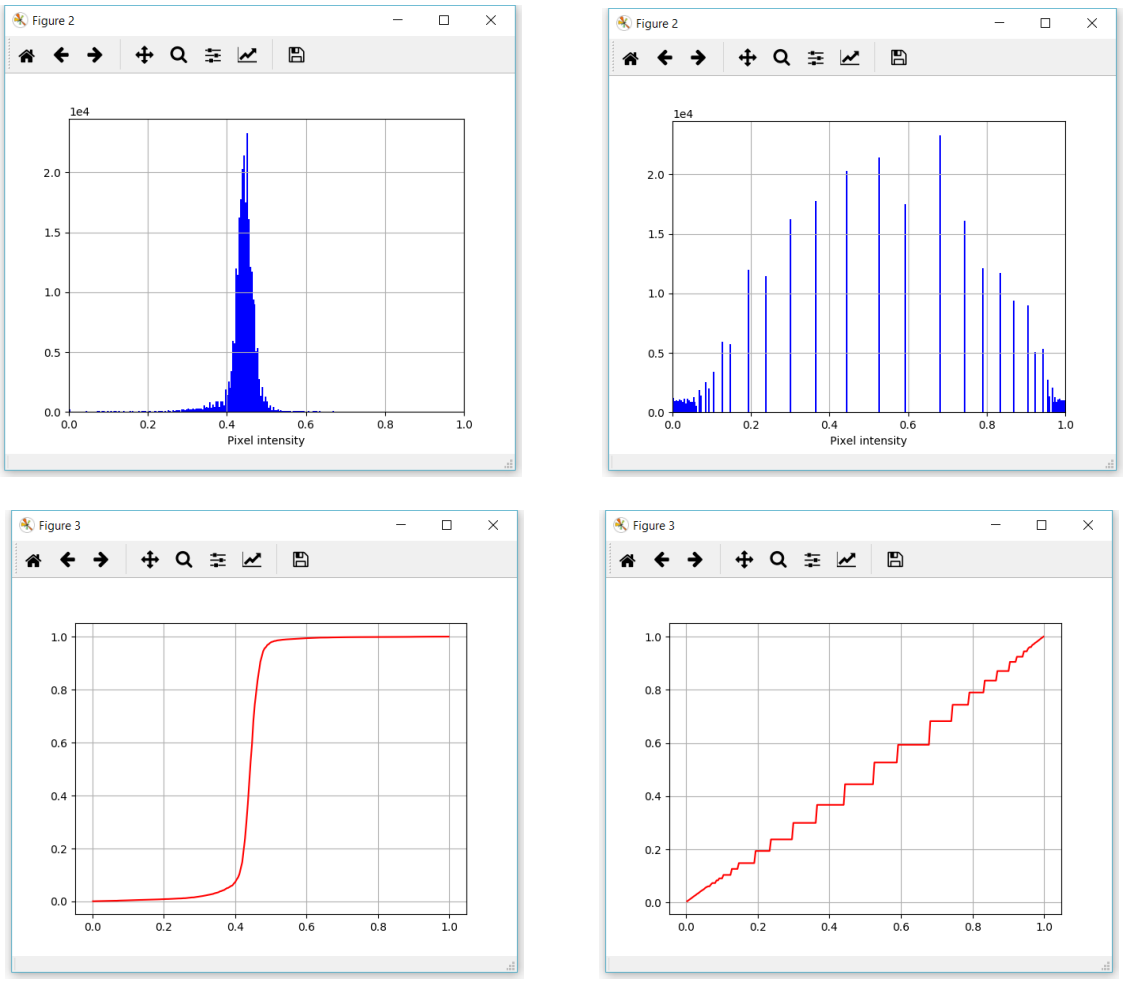
e)

Rys. 6. a) Obraz testowy skimage.data.moon() w oryginale i po rozciągnięciu histogramu umieszczone w jednym oknie, b) histogram oryginału, c) histogram rozciągnięty, d) dystrybuanta oryginału, e) dystrybuanta po rozciągnięciu

Zadanie 6

Napisać funkcję contrast_equalize do wyrównywania kontrastu w obrazie 2D z wieloma poziomami jasności, wyświetlającą obraz po załadowaniu i obraz wynikowy z wyrównanym kontrastem (matplotlib.pyplot.hist). Wyświetlić także histogram obrazu przed i po rozciągnięciu dla różnej ilości słupków (32, 64, 256) . Zastosować funkcję wbudowaną (skimage.exposure.equalize_hist, skimage. exposure.equalize_adapthist oraz cv2. equalizeHist).





Rys. 7. a) Obraz testowy skimage.data.moon() w oryginale i po globalnym wyrównaniu histogramu umieszczone w jednym oknie, b) histogram oryginału, c) histogram wyrównany, d) dystrybuanta oryginału, e) dystrybuanta po wyrównaniu