# PROJEKT I

Krzysztof Kowalski 407142

Celem projektu było wykonanie 4 przekształceń/funkcji zaimplementowanych w Pythonie, które wykonują 4 różne przekształcenia związane z przetwarzaniem i analizą obrazów. W moim przypadku będą to:

- odpowiednik MatLab'owego **regionprops** wyliczająca centroidy oraz ekwiwalent średnic (ilość pikseli = pole koła), dla obrazu monochromatycznego

- odpowiednik MatLab'owego **ordfilt2** dla zadanego rozmiaru maski i zadanego numeru porządkowego. Dla obrazu monochromatycznego oraz RGB (każda warstwa osobno)

- odpowiednik MatLab'owego **imopen**, a zatem otwarcie morfologiczne elementem kołowym o zadanym promieniu

- odpowiednik MatLab'owego **imfill(obraz, 'holes')**, a zatem wypełnianie dziur w obiektach przez rekonstrukcję.

**1. Odpowiednik regionprops**

Funkcja ta w MatLab'ie służy do obliczania różnych właściwości znalezionych na obrazie monochromatycznym obiektów. Ogrom tych cech można następnie zastosować do detekcji obiektów, czy wielu innych operacji związanych z analizą obrazów.

W Pythonie zaimplementowana została funkcja regionsprops, przyjmująca dwa argument: ścieżkę do obrazu, który ma zostać poddany analizie oraz ścieżkę gdzie wyliczone wartości centriodów oraz ekwiwalentów średnic w postaci pliku tekstowego mają zostać zapisane. Algorytm wczytuje obraz ze ścieżki. Następnie inicjalizowana jest maksymalna ilość iteracji (poziomów szarości) oraz słownik odpowiadająca za centroidy i listę odpowiadająca za ekwiwalent średnicy. Następnie wszystkie wartości mają przypisaną wartość 0. Kolejnym krokiem jest obliczenie ilości pikseli dla każdego poziomu szarości. Następnie liczona jest średnia wartość współrzędnych dla każdego poziomu szarości oraz obliczania jest średnica ekwiwalentna na podstawie liczby pikseli. Ostanim etapem funkcji jest zapisywanie wyliczonych wielkości do pliku tekstowego.

Wynik działania dla obrazu monochromatycznego 'cameraman.tif'

```
Nr,Centroids,EquivalentDiameters
0,-1.00,-1.00,0.00
1,-1.00,-1.00,0.00
2,-1.00,-1.00,0.00
3,-1.00,-1.00,0.00
4,-1.00,-1.00,0.00
5,-1.00,-1.00,0.00
6,-1.00,-1.00,0.00
7,62.25,134.50,2.26
8,93.40,132.64,23.21
9,93.44,132.32,43.37
10,95.29,135.05,40.04
11,83.49,135.89,38.68
12,74.91,140.66,43.06
13,76.06,144.49,44.12
14,79.22,151.56,46.32
15,81.47,146.62,41.27
16,79.58,142.88,35.11
17,83.17,139.24,24.49
18,88.75,142.08,18.23
19,96.75,146.28,15.43
20,100.36,150.42,14.67
21,89.16,173.57,13.35
22,104.32,155.48,11.67
23,105.17,164.83,11.78
24,94.73,183.50,11.51
25,109.88,173.36,11.06
26,113.78,164.39,11.23
27,120.21,156.18,12.05
28,118.22,179.80,10.46
29,117.56,168.83,11.73
```

**2. Odpowiednik ordfilt2**

Funkcja ta w MatLab'ie służy do filtorwania obrazu przy użyciu filtra porządkowego (jest to filtracja nieliniowa). Filtracja ta zamienia wartości pikseli w obszarze maski na wartości, które znajdują się na określonej pozycji w posortowanej masce.

W Pythonie zaimplementowana została funkcja ordfilt2_mono oraz ordfilt2_color. Obie przyjmują taka samą ilość argumentów, które są kolejno scieżka do obrazu, który ma zostać przetworzony, ranga (pozycja) oraz rozmiar maski. Ordfilt_2_mono po wczytaniu obrazu szarego ze ścieżki tworzy zmienne przechowujące wymiary obrazu, marginesy dla maski oraz tablice wynikową. Następnie w dwóch pętlach for przechodząc po długości i szerokości obrazu, a zatem przechodząc po każdym pikselu tworzona jest maska poprzez przycięcie

obrazu do rozmiaru maski, a następnie sortuje wartości pikseli w tej masce. Ostatnim krokiem przed zapisaniem obrazu jest wstawienie za aktualne współrzędne piksela wartości z posortowanej maski o indeksie podanej jako drugi parametr rangi. Funkcja ordfilt_2_color działa analogicznie. Jedyną różnicą jest, że oprócz wymiarów obrazu, przypisywana do zmiennej jest też ilość warstw. Występuje tutaj potrójna pętla for, która przechodzi po każdym pikselu w kolejnych warstwach i wykonuję dla pojedyncze warstwy analogiczne obliczenia co ordfilt2_mono. Wynik działania algorytmu zapisywany jest jako nowy obraz RGB.

Wynik działania dla obrazu monochromatycznego ‘cameraman.tif’, rozmiar maski 12, numer porządkowy 5

Wynik działania dla obrazu kolorowego ‘pears.png’, rozmiar maski 12, numer porządkowy 5

**3. Odpowiednik imopen**

Funkcja ta w MatLab’ie służy do wykonywania otwarcia morfologicznego na obrazie. Otwarcie to jedna z podstawowych operacji morfologicznych, która polega na wykonaniu na obrazie sekwencji dwóch operacji: erozji, a następnie dylacji.

W Pythonie zaimplementowana została funkcja imopen, która korzysta pomocniczo z funkcji circle_crate, dilate oraz erode.

Funkcja creat_circle odpowiada za stworzenie elementu strukturalnego w kształcie koła o podanym promieniu. Tworzymy w niej zmienne które odpowiadają za środek macierzy następnie w dwóch pętlach for przechodzimy po wszystkich pikselach macierzy, tak aby ten element miał kształt koła, a zatem obliczamy odległość od środka i jeśli jest ona mniejsza lub równa promieniowi to ustawiamy wartość piksela na 1.

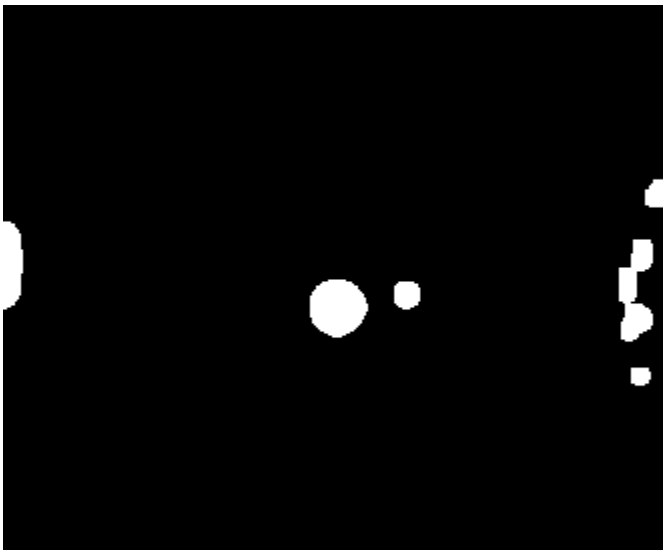
Kolejna pomocnicza funkcja erode, odpowiada za wykonanie erozji morfologicznej. W funkcji tej tworzymy kopię oryginalnego obrazu a następnie w dwóch pętlach for przechodząc po wszystkich pikselach tworzymy nowa zmienna mająca maksymalną wartość dla piksela równa 255. Następnie badamy czy piksel jest w obrębie elementu strukturalnego, jeśli tak to zamieniamy go na minimum z wcześniej utworzonej zmiennej i piksela, a w kopii oryginalnego obrazu zamieniamy aktualny piksel na wartość z tego minimum. Zwracana jest przetworzona kopia oryginalnego obrazu.

Kolejna pomocnicza funkcja jest dilate. Jej działanie jest analogiczne do funkcji erode z tą różnicą że jeśli warunek jest spełniony to aktualizujemy piksel na maksimum, a nie minimum. Tutaj również zwracana jest zaktualizowana kopia oryginalnego obrazu.

Funkcja imopen odpowiada za wykorzystanie wszystkich wyżej opisanych funkcji pomocniczych. Początkowo funkcja wczytuje obraz ze ścieżki, następnie tworzy pomocniczą macierz potrzebną do stworzenia elementu strukturalnego. Następnie do tak zaincjalizowanej macierzy przypisywany jest wynik działania funkcji create_circles z odpowiednimi parametrami (stworzony zostaje element strukturalny w kształcie koła). Zostaje już tylko wykonanie funkcji erode, a następnie dilate na wyniku erozji. Ostatecznie zapisywany jest w podanej jako parametr ścieżce przetworzony obraz.

Wynik działania dla obrazu monochromatycznego ‘cameraman.tif’ o promieniu 5

Wynik działania dla obrazu logicznego ‘blobs.png’ o promieniu 5



**4. Odpowiednik imfill(obraz, ‘holes’)**

Funkcja ta w MatLab’ie odpowiada za wypełnianie dziur w obiektach w obrazach logicznych (ma ona więcej zastosowań, ale po podaniu drugiego argumentu jako ‘holes’ zadziała jako wypełnianie dziur w obiektach).

W Pythonie zaimplementowana została funkcja imfill, która pomocniczo korzysta z funkcji dilation oraz max. Funkcja dilation wykonuje operację dylacji na obrazie binarnym podanym jako argument z użyciem maski (również podanej jako parametr). Tworzona jest zwracana później macierz będąca kopią wejściowego obrazu. Następnie w dwóch pętlach for przechodząc po każdym pikselu wywoływana jest funkcja max, aby określić wartość maksymalna dla piksela oraz maski. Wynik działania funkcji maksymalizującej przypisywany jest do stworzonej przed pętlą kopii za odpowiedni piksel. Zwracana jest zmodyfikowana kopia obrazu wejściowego, która reprezentuje wynik dylacji.

Kolejną funkcją pomocniczą jest max. Funkcja ta oblicza wartość maksymalną dla danego piksela obrazu na podstawie maski. Przechodząc przez otoczenie piksela (zależne od promienia), sprawdzane są warunki czy piksel mieści się w granicach obrazu oraz warunek odległości od piksela (podanego jako argument funkcji). Jeśli warunki są spełnione i piksel w masce ma wartość jeden, zwracana jest prawda, w przeciwnym przypadku zwracany jest fałsz.

Funkcja imfill, która służy do wypełniania dziur w obrazie binarnym. Na początku funkcji tworzona jest maska (macierz zer o rozmiarze obrazu wejściowego). Następnie w dwóch pętlach for piksele znajdujące się na brzegu obrazu dostają wartość prawdy. Kolejnym krokiem jest negacja obrazu wejściowego. W nieskończonej pętli, której warunkiem stopu jest gdy nie ma różnicy pomiędzy wynikiem dylacji a maską (dylacja wykonywana jest w każdej iteracji pętli), na wyniku, którym wykonywana jest operacja koniunkcji z negacją obrazu wejściowego. Jeśli warunek stopu został spełniony następuje przypisanie do wyjściowego pliku negacji wyniku pomnożonego przez 255 (w celu uzyskania zakresu 0 - 255).

Wynik działania dla obrazu logicznego ‘dziury.bmp’ o promieniu 5

**INSTRUKCJA DZIAŁANIA**

Należy uruchomić plik **main.py** (pliki **regionprops.py, ordfilt2.py, imopen.py, imfill.py** muszą znajdować się w tym samym elemencie ścieżki co main.py. Po odpaleniu należy postępować z instrukcjami wyświetlającymi się w konsoli, czyli podać ścieżkę gdzie znajduje się obraz, który ma być poddany analizie, ewentualne parametry jeśli funkcja tego wymaga oraz ścieżkę wraz z rozszerzeniem gdzie przetworzony plik ma być zapisany.