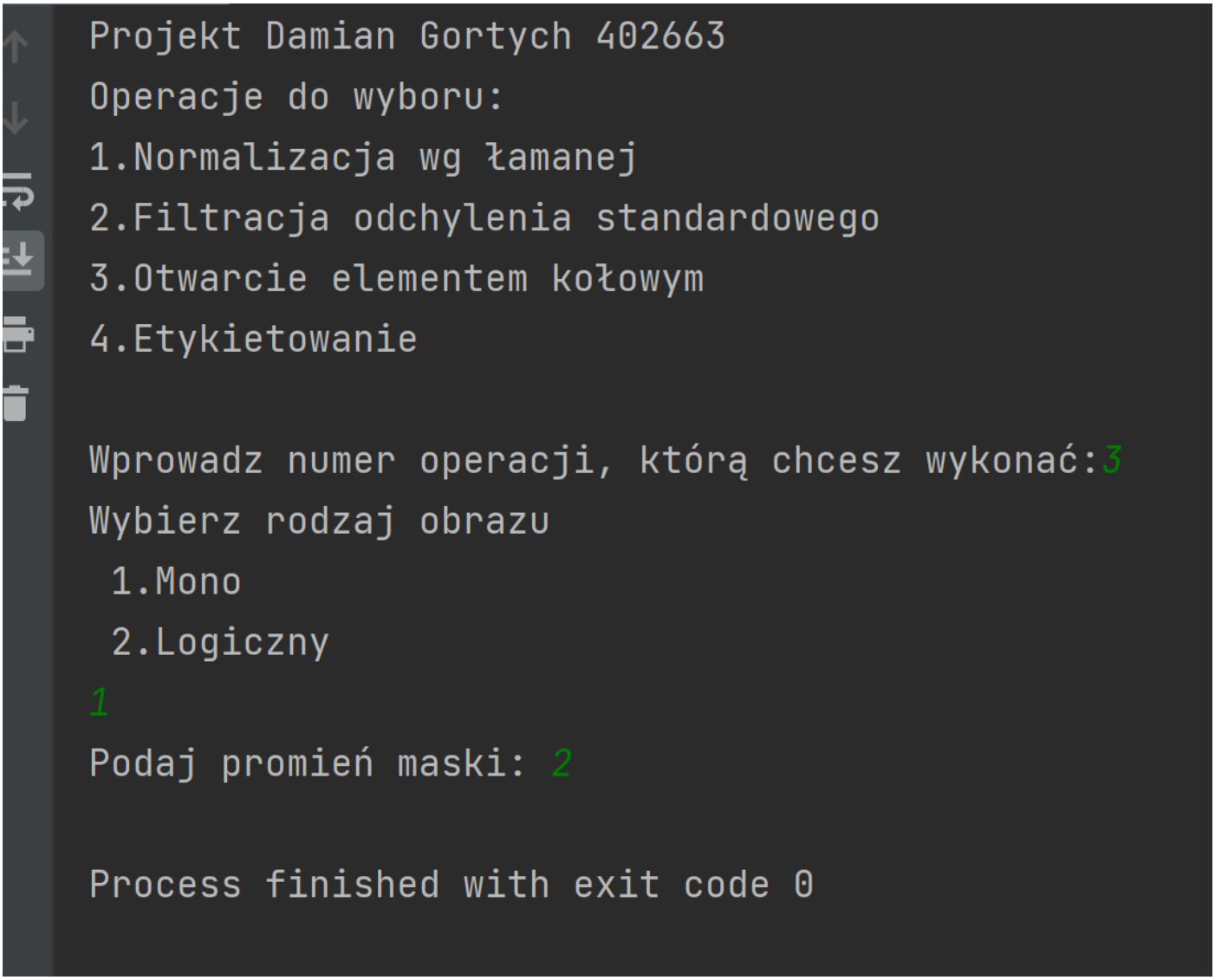
Raport Projekt 1 Damian Gortych 402663

Projekt wykonałem w języku Python.

Składa się on z 4 modułów, w których wykonywane są przekształcenia dla poszczególnych operacji oraz modułu main, w którym znajduje się proste menu w którym użytkownik może wybierać rodzaj operacji, typ obrazu oraz zadane parametry.

Przykład menu dla otwarcia :



Na początku chciałbym zaznaczyć, iż w całym projekcie jako wartości **True** oraz **False** (np. w przypadku obrazów logicznych) używałem wartości **0** (False) i **255** (True), ponieważ to ułatwiało mi operacje, a nie wpływało na ostateczny wynik przekształcenia.

W projekcie użyłem biblioteki **pillow** do otwarcia oraz zapisu zdjęcia oraz biblioteki **numpy** do prostych przekształceń na macierzy. Większość nawet wbudowanych już funkcji napisałem samemu. Posłużyłem się jedynie funkcją pad(), która dodaje obramowanie wokół macierzy.

1. Normalizacja wg łamanej

Dla tej operacji wybrałem obraz **white_tern.bmp** jako RGB oraz **cameraman.png** jako monochromatyczny

Opis działania:

1. Wczytanie obrazu
2. Wczytanie punktów od użytkownika
3. W pętli dla każdej pary punktów następuje przechodzenie po obrazie i każdy pixel jest normalizowany do danego przedziału wyjściowego jeśli znajduje się w danym przedziale wejściowym.
4. Zapis obrazu wyjściowego

W przypadku obrazu RGB każdy z kolorów jest normalizowany osobno tak jak dla obrazu monochromatycznego.

Przykład wyników dla 3 punktów (50,100), (100,160), (170,220)

Obraz wejściowy RGB:

Obraz wyjściowy RGB:

Obraz wejściowy monochromatyczny:

Obraz wyjściowy monochromatyczny:

2. Filtracja odchylenia standardowego

Dla tej operacji wybrałem obraz **white_tern.bmp** jako RGB oraz **cameraman.png** jako monochromatyczny.

Opis działania:

1. Wczytanie obrazu
2. Stworzenie maski wg zadanych parametrów
3. Filtracja polegająca na liczeniu odchylenia standardowego wg maski
4. Normalizacja

Przykład wyników dla maski 5x5:

Obraz wejściowy RGB:

Obraz wyjściowy RGB:

Obraz wejściowy monochromatyczny:

Obraz wyjściowy monochromatyczny:

3. Otwarcie elementem kołowym

Dla tej operacji wybrałem obraz **blobs.png** jako logiczny oraz **cameraman.png** jako monochromatyczny.

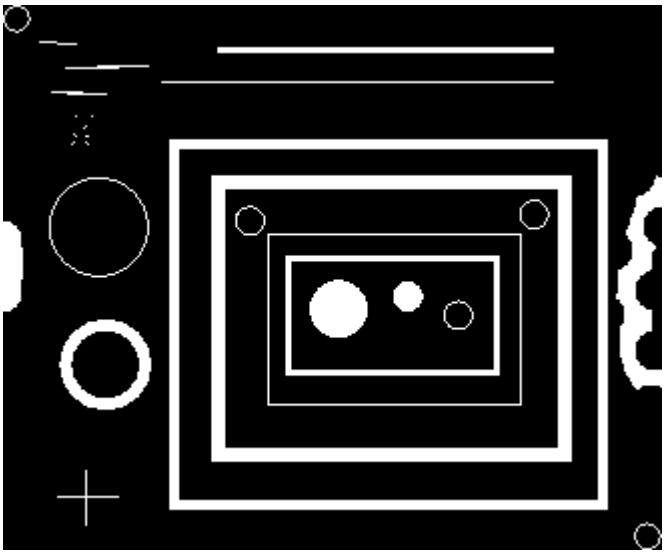
Opis działania:

1. Wczytanie obrazu
2. Stworzenie kołowego elementu strukturalnego wg zadanego promienia
3. Otwarcie poprzez wykonanie erozji a następnie dylacji.
4. Zapisanie obrazu wyjściowego

Pomiędzy obrazami w procesie główna różnica występuje w etapach erozji oraz dylacji, ponieważ dla obrazu monochromatycznego szukamy minimum lub maximum w zadanym sąsiedztwie.

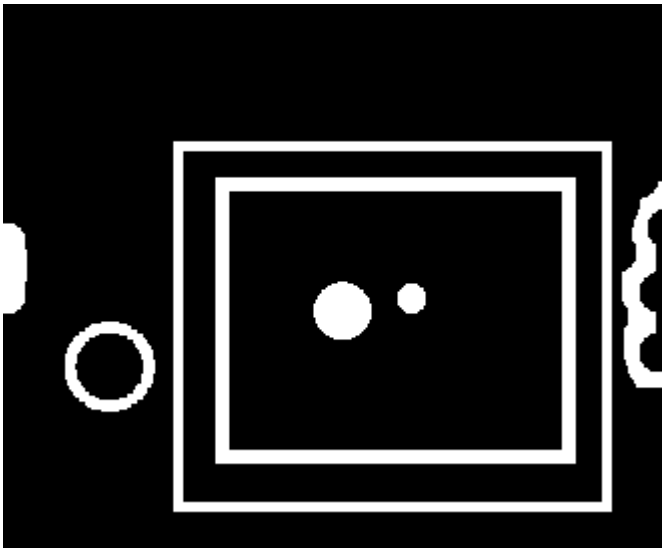
Przykład wyników dla maski o promieniu 3 :

Obraz wejściowy logiczny:



Obraz wejściowy monochromatyczny:

Obraz wyjściowy logiczny:



Obraz wyjściowy monochromatyczny:

4. Etykietowanie

Dla tej operacji wybrałem **zbinaryzowany** obraz **coins.png**

Opis działania:

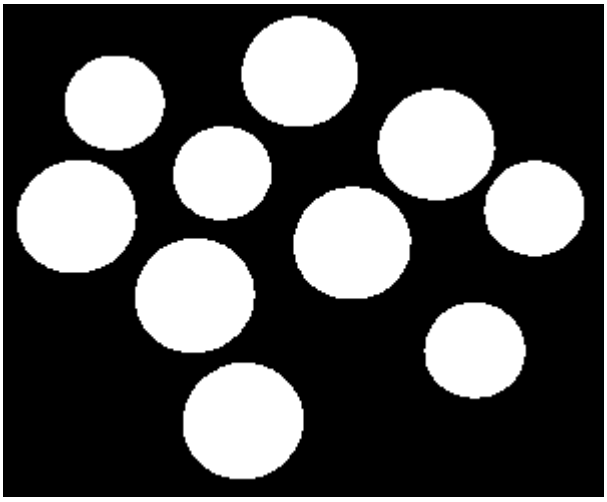
1. Wczytanie obrazu
2. Przechodzenie w pętli po obrazie kolumnami
3. W przypadku znalezienia pixela z wartością 255 następuje rekonstrukcja morfologiczna w której tworzony jest marker i w nim odwzorowywany jest element
4. Na podstawie znalezionego elementu w markerze wykonywane jest przypisanie etykiety do obrazu wejściowego
5. Zwiększenie etykiety o 1, dalsze wykonywanie w pętli, aż do znalezienia wszystkich elementów.

W procesie nie jest wykonywane odejmowanie znalezionego elementu od obrazu wejściowego, ponieważ przy przechodzeniu w pętli sprawdzana jest wartość pixela 255, a pixele już zaetykietowane mają wartości 1,2,3 …

Jednakże kod funkcji i funkcjonalność została zaimplementowana i jest dostępna do użycia w razie chęci zmian w przyszłości.

W celu lepszego pokazania działania operacji obraz wyjściowy pomnożyłem razy 20 (co oznacza etykiety 20, 40 ,60 ..), aby wynik był bardziej widoczny.

Obraz wejściowy:



Obraz wyjściowy: