
ANALIZA I PRZETWARZANIE OBRAZÓW CYFROWYCH

SPRAWOZDANIE – PROJEKT I

Celem projektu była implementacja wybranych przekształceń na obrazach, pogłębiając przy tym wiedzę na temat sposobu działania danych operacji.

Zostały mi przypisane do przetestowania następujące przekształcenia:

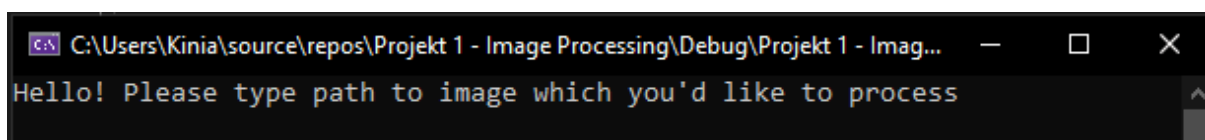
1. Przekształcenie Afiniczne
2. Filtracja Odchylenia Standardowego w zadanej masce
3. Zamknięcie elementem liniowym o zadanej długości i nachyleniu
4. Etykietowanie

Program został napisany w języku C++. Jest to aplikacja CLR we frameworku .NETv4.5.2. Do operacji odczytu, zapisu, alokacji oraz wyświetlania użyta została przestrzeń nazw System::Drawing. Całość zaimplementowana, uruchamiana i testowana w środowisku Visual Studio 2019.

Uruchamianie programu

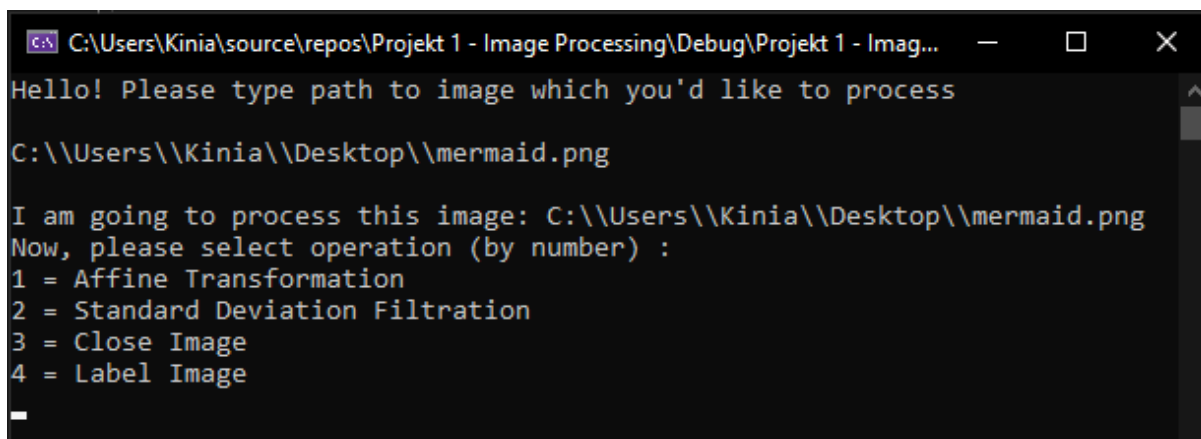
Po uruchomieniu zostajemy poproszeni o wskazanie ścieżki do pliku. Powinna ona być w formacie:

C:\\Users\\Kinia\\Desktop\\mermaid.png



```
C:\Users\Kinia\source\repos\Projekt 1 - Image Processing\Debug\Projekt 1 - Imag...
Hello! Please type path to image which you'd like to process
```

Po wpisaniu ścieżki należy wybrać operację którą chcemy zastosować na danym obrazie.



```
C:\Users\Kinia\source\repos\Projekt 1 - Image Processing\Debug\Projekt 1 - Imag...
Hello! Please type path to image which you'd like to process
C:\\Users\\Kinia\\Desktop\\mermaid.png

I am going to process this image: C:\\Users\\Kinia\\Desktop\\mermaid.png
Now, please select operation (by number) :
1 = Affine Transformation
2 = Standard Deviation Filtration
3 = Close Image
4 = Label Image
_
```

Przekształcenie Afiniczne

Transformacja afiniczna jest przekształceniem geometrycznym. Należą do niej operacje takie jak : translacja, skalowanie, odbicie, obrót, pochylenie, a także ich dowolne złożenie.

Mój program umożliwia translację, skalowanie, obrót, pochylenie oraz przekształcenie poprzez macierz transformacji . Testowane na obrazach RGB oraz monochromatycznych. Wykorzystuje interpolację – poprzez nadanie sąsiednim pikselom wartości aktualnego - by zapobiec 'dziurom' między pikselami.

```
C:\Users\Kinia\source\repos\Projekt 1 - Image Processing\Debug\Projekt 1 - Image Processi...
Hello! Please type path to image which you'd like to process

C:\\Users\\Kinia\\Desktop\\mermaid.png

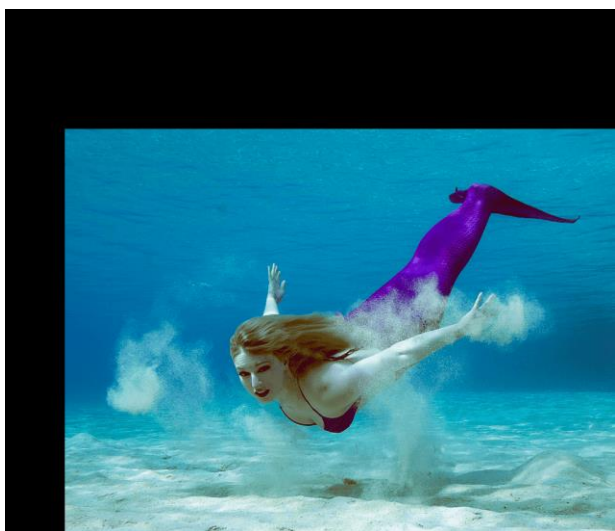
I am going to process this image: C:\\Users\\Kinia\\Desktop\\mermaid.png
Now, please select operation (by number) :
1 = Affine Transformation
2 = Standard Deviation Filtration
3 = Close Image
4 = Label Image
1

You choosed 1

Choose Affine Transformation
1 - Translate
2 - Scaling
3 - Rotation
4 - Shear
5 - By Matrix
-
```

Translacja – oznacza przesunięcie obrazu o zadaną wysokość i szerokość. Obraz wynikowy zostaje poszerzony o podane parametry, by nie utracić żadnej części oryginalnego obrazu. Do współrzędnej każdego piksela są dodawane podane wartości.

Przykładowe wyniki przesunięcia o 100, 200:



Skalowanie – zmiana rozmiaru poprzez zadany współczynnik. Uwzględniane są dwa współczynniki odnoszące się do skali w poziomie i pionie. Każdy piksel zostaje pomnożony przez określoną wartość.

(przykładowe obrazki poniżej pomniejszone na rzecz oszczędności miejsca na stronie.)

Skalowanie o 1, 0.5:



2, 0.7:



Rotacja – Obrót o wskazany kąt. Wyjściowy poszerzony rozmiar obrazka jest obliczony z pomocą własności trójkąta prostokątnego. Nowe współrzędne piksela są obliczane zgodnie z równaniem:

$$\begin{aligned} \text{posx} &= x * \cos\alpha - y * \sin\alpha + x_0 \\ \text{posy} &= x * \sin\alpha - y * \cos\alpha + y_0 \end{aligned}$$

90 stopni:



38 stopni:



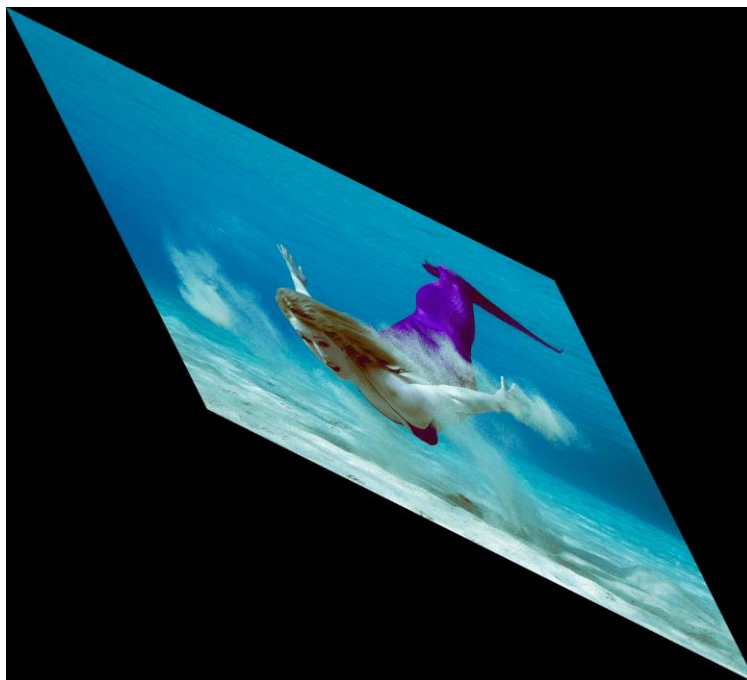
Pochylenie – o zadane współczynniki. 0 oznacza brak pochylenia. Nowe współrzędne są obliczane z pomocą równań :

$$\text{posx} = x + a*y$$

$$\text{posy} = y + b*x$$

Przykładowe wyniki:

0.5,0.5:



0.6, 0:



Poprzez **macierz transformacji**, rozmiaru 2x3. Program prosi o wpisanie poszczególnych elementów macierzy. Nowa pozycja pikseli jest wyznaczana poprzez mnożenie wektora współrzędnych i macierzy:

$$\begin{bmatrix} \text{posx} & \text{posy} \end{bmatrix} = \begin{bmatrix} x & y \end{bmatrix} * \begin{bmatrix} a & b & c & d & e & f \end{bmatrix}$$

$$\text{posx} = x*a + y*c + e$$

$$\text{posy} = x*b + y*d + f$$

Gdzie posx, posy oznaczają nowe położenie pikseli, x,y współrzędne na oryginalnym obrazie, a-f elementy zadanej macierzy.

Przykładowe przekształcenie macierzą

$$\begin{bmatrix} 1 & 0.5 \\ 0 & 2 \\ 0 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 1 & 0 \\ 0.3 & 1 \\ 50 & 50 \end{bmatrix}$$



Filtracja Odchylenia Standardowego

Początkowo pobieramy rozmiar maski, a obraz jest poszerzany o ten rozmiar (by upłynnić obliczenia dla skrajnych pikseli) . Dla każdego piksela obliczamy wartość odchylenia standardowego uwzględniając piksele z jego otoczenia(wielkości zadanej maski). Odchylenie jest obliczane dla każdej z wartości R, G, B osobno. Wyniki są poddawane normalizacji by zapobiec wykroczenia poza zakres 0-255. W przypadku wartości <0, przypisywane jest 0, natomiast przy >255 – analogicznie 255.

Odchylenie jest obliczane wedle poniższego wzoru :

$$stddev = \sqrt{\frac{\sum (x_i - \bar{x})^2}{r * c - 1}}$$

Gdzie x_i oznacza rozważany aktualnie piksel, \bar{x} – średnią wszystkich branych pod uwagę pikseli z jego otoczenia, natomiast r, c – szerokość oraz wysokość.

Oto przykładowe wyniki filtracji:

Obraz oryginalny:



Po filtracji maską o rozmiarze 3:



Obraz oryginalny:



Po filtracji maska rozmiar 7:



Obraz monochromatyczny :

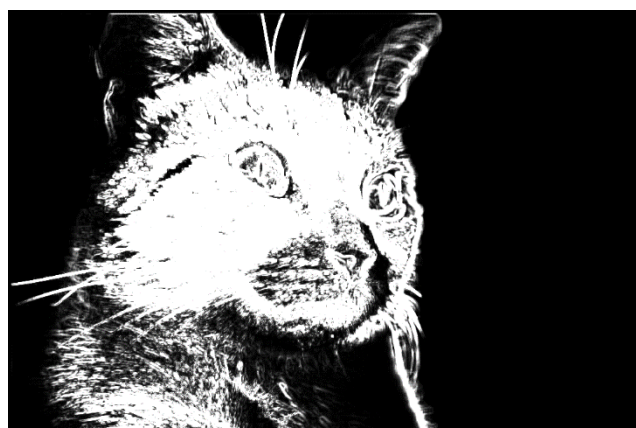
obraz oryginalny:



maska rozmiar 3:



maska rozmiar 5:



Zamknięcie Obrazu

Zamknięciem obrazu nazywamy złożenie dwóch operacji – erozji oraz dylacji. W pierwszej kolejności wykonywana jest dylacja, następnie przekształcony nią obraz poddajemy erozji. Dane mi było przetestować zamknięcie elementem liniowym. Na starcie po wybraniu operacji podajemy programowi wybrane parametry – długość oraz nachylenie elementu (w granicach 0- 180stopni).

Rozmiar elementu liniowego wyciągamy z pomocą własności kątów i boków.

```
rows = abs(ceil(sin(rad)* Length));  
columns = abs(ceil(cos(rad)* Length));
```

Element liniowy tworzymy na podstawie równania prostej $y=a*x + b$. Współczynnik a jest obliczany poprzez wzór $a = \text{tangens}(a)$. Następnie każdemu elementowi macierzy który „przecina się” z naszą prostą, przypisujemy wartość 1. Przy kącie większym niż 90stopni, odbijamy tę prostą względem osi x, to znaczy: zamiast przypisać wartości jeden w elemencie (y, x), przypisujemy ją w elemencie (y, szerokość -x -1).

Element liniowy prezentuje jednowymiarowa tablica. Do wyciągania wartości danego wiersza i kolumny została zaimplementowana funkcja *sub2ind*.

Przykładowe elementy liniowe:

```
Enter size: 10  
Enter angle (0-180) :135  
8x8  
1 0 0 0 0 0 0 0  
0 1 0 0 0 0 0 0  
0 0 1 0 0 0 0 0  
0 0 0 1 0 0 0 0  
0 0 0 0 1 0 0 0  
0 0 0 0 0 1 0 0  
0 0 0 0 0 0 1 0  
0 0 0 0 0 0 0 1
```

```
Enter size: 4  
Enter angle (0-180) :90  
4x1  
1  
1  
1  
1
```

```
Enter size: 5  
Enter angle (0-180) :30  
3x5  
0 0 0 0 1  
0 1 1 1 0  
1 1 0 0 0
```

Erozja i Dylacja należą do morfologii matematycznej, operują na elementach true/false.

Dylacja – polega na zwiększeniu/rozszerzaniu obiektów. Jest zaimplementowana w funkcji *Dilate()*. Dla danego piksela pobiera wszystkie piksele z jego otoczenia, rozmiaru zadanej maski, oznaczone '1' i wybiera wartość maksymalną.

Erozja – przeciwnie do dylacji, ma na celu pomniejszenie obiektów. Implementacja (funkcja *Erode()*) analogiczna: dla każdego piksela w obrębie maski oznaczonego '1', wybiera wartość minimalną i nadaje ją pikselowi o tych samych współrzędnych w obrazie wyjściowym.

Program prosi nas o określenie obrazu jaki pragniemy przetworzyć – w przypadku obrazu monochromatycznego początkowo jest on poddany binaryzacji o progu 150.

Przykładowe operacje zamknięcia:

Obraz binarny :



długość 5, nachylenie 135:



długość 10, nachylenie 45:



Obraz monochromatyczny:



długość 10, nachylenie 135:



długość 10, nachylenie 30:



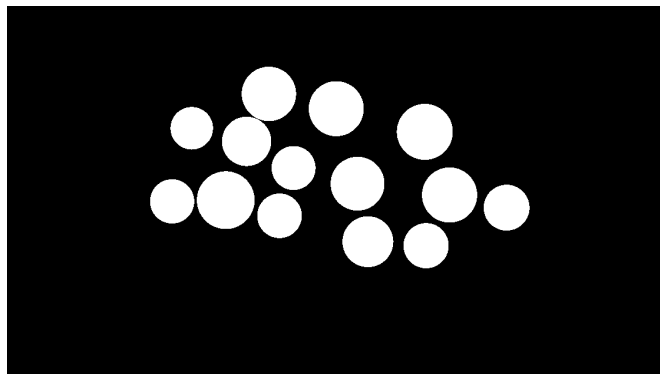
Etykietowanie

Operacja etykietowania polega na rozdzieleniu obiektów na obrazie i przypisania im indeksu, etykiety.

W moim programie obraz jest początkowo przekształcany w macierz 0 i 1. 0 oznacza tło obrazu, natomiast 1 – część obiektu. Następnie w utworzonej macierzy przeprowadzam podział z wykorzystaniem rekursji. Pierwszą przypisywaną etykietą jest nr2, by odróżnić etykiety od 0 i 1. Dla każdego piksela który jest nieoznaczony i nie jest tłem (czyli jest równy 1), sprawdzamy piksele z jego otoczenia. Jeśli również spełniają powyższe warunki, nadawana jest im bieżąca etykieta. Jest to algorytm etykietowania poprzez rozrost obszaru. Dla każdego kolejnego piksela który spełnia warunki są sprawdzani jego sąsiedzi. Po przejściu całego obszaru, zwiększamy numer etykiety i szukamy kolejnych nieoznaczonych obiektów.

By zaprezentować efekt etykietowania możemy wypisać macierz funkcją *Printlabel()*, jednak jej wielkość nie pozwala na szybką analizę poprawności algorytmu, dlatego postanowiłam zapisać poszczególne elementy obrazu – każdy obiekt do osobnego pliku – obsługuje to funkcja *SaveSeparatedAreas()* na podstawie numeru etykiety.

Przykład :



Folder wynikowy, po operacji etykietowania:

