

Biometria - projekt 1

Aplikacja do przetwarzania obrazów
Wydział Matematyki i Nauk Informacyjnych

Paweł Florek

Marzec 2025

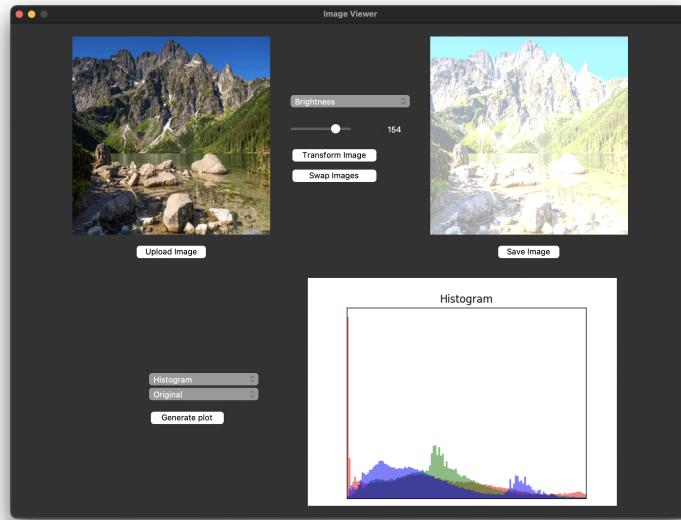
Spis treści

1 Wstęp	3
2 Filtry	3
2.1 Grayscale	4
2.2 Binaryzacja obrazu	5
2.3 Korekcja jasności obrazu	5
2.4 Korekcja kontrastu obrazu	6
2.5 Filtr negatywu obrazu	7
2.6 Binaryzacja obrazu w 3 kolorach	8
2.7 Filtr uśredniający	9
2.8 Filtr medianowy	10
2.9 Filtr Kuwahary	11
2.10 Filtr Gaussa	12
2.11 Filtr wyostrzający	13
2.12 Operator Robertsa	14
2.13 Operator Sobela	15
2.14 Filtr górnoprzepustowy	16
2.15 Operator Laplace'a	17
2.16 Operator Prewitta	18
2.17 Filtr Ridge	19
2.18 Operator Scharra	20
3 Wykresy	22
3.1 Histogram	22
3.2 Projekcja Pozioma	23
3.3 Projekcja Pionowa	24
4 Inne funkcjonalności	25
5 Podsumowanie	26

1 Wstęp

Opisana poniżej aplikacja umożliwia stosowanie podstawowych filtrów na wczytanych obrazach. Aplikacja pozwala na importowanie oraz zapisywanie obrazów, a także zastosowanie jednego z dostępnych filtrów, po czym użytkownik ma możliwość zapisania przetworzonego obrazu lub kontynuowania pracy z kolejnymi filtrami. Dodatkowo, aplikacja oferuje funkcje analizy obrazu za pomocą histogramu oraz projekcji pionowej i poziomej.

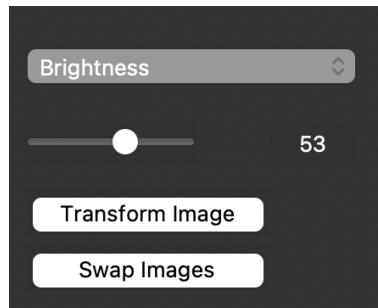
Aplikacja została opracowana z wykorzystaniem biblioteki Tkinter, która zapewnia interfejs graficzny. Filtry obrazów są implementowane przy pomocy biblioteki NumPy, natomiast operacje związane z wczytywaniem i zapisywaniem obrazów wspierane są przez bibliotekę Pillow.



Rysunek 1: GUI aplikacji

2 Filtry

Filtr można wybrać za pomocą rozwijanego menu (dropdown menu), a następnie, w przypadku większości filtrów, dostosować jego moc lub wielkość jądra przy użyciu suwaka (slidera). W przypadku filtrów, które nie oferują opcji personalizacji, suwak jest wyłączony i wyszarzony, uniemożliwiając jego użycie oraz podana wartość nie wpływa na działanie filtru. Po zastosowaniu wybranego filtru użytkownik ma możliwość zamiany obrazów, co pozwala na dalszą edycję i eksperymentowanie z kolejnymi filtrami.

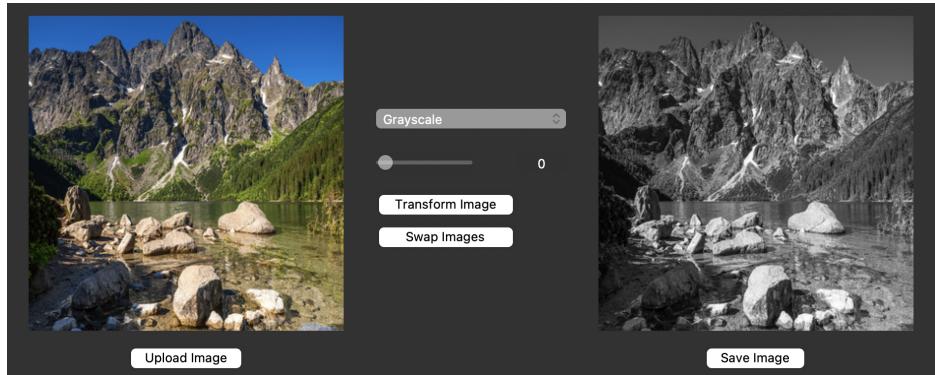


Rysunek 2: Opcje filtra

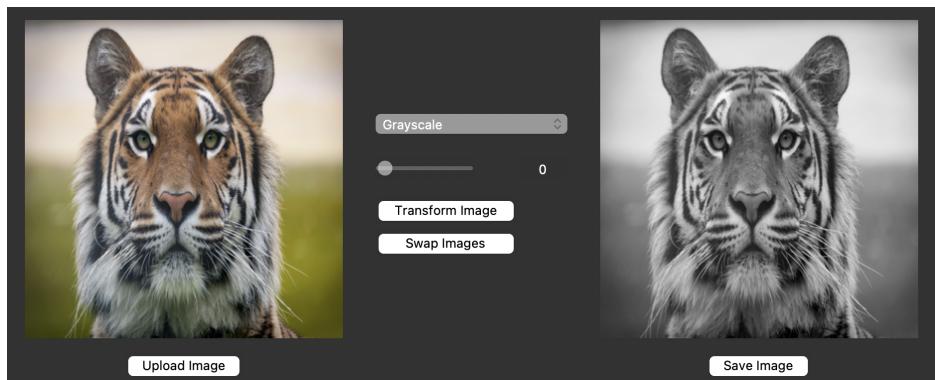
2.1 Grayscale

- `def grayscale(image),`
- `image` - obrazek poddawany filtrowaniu
- Zastosowanie - konwersja obrazka w obraz monochromatyczny,
- Transformacja - dla każdego piksela $I(x, y) = (R, G, B)$ w obrazie:

$$I'(x, y) = 0.2989 \cdot R + 0.5870 \cdot G + 0.1140 \cdot B$$



Rysunek 3: Grayscale nr 1



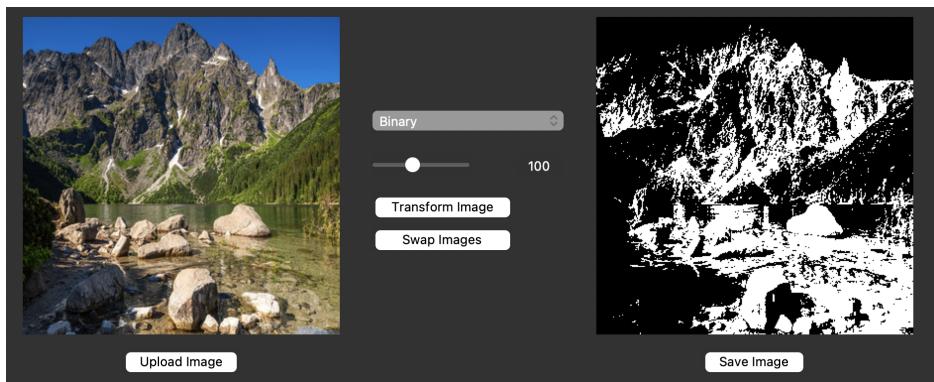
Rysunek 4: Grayscale nr 2

2.2 Binaryzacja obrazu

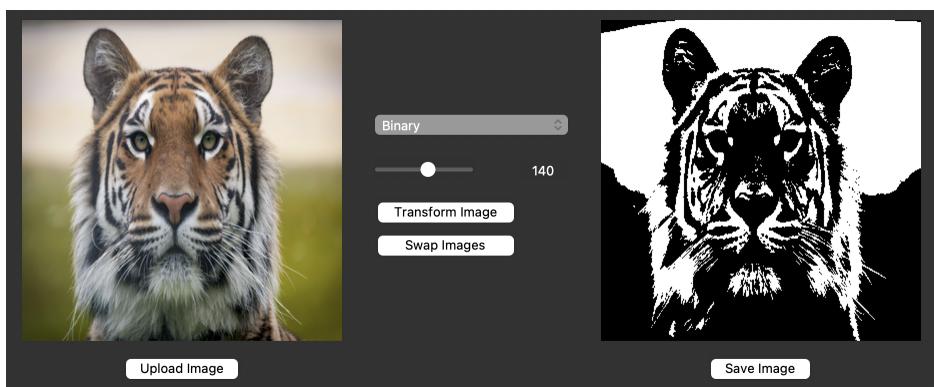
- `def binary(image, threshold),`
- `image` - obrazek poddawany filtrowaniu,
- `threshold` - próg, powyżej którego piksele stają się białe (255), poniżej czarne (0),
- Zastosowanie - konwersja obrazu do wersji binarnej, w której piksele są tylko czarne lub białe, w zależności od progu,
- Transformacja - dla każdego piksela $I(x, y)$ w obrazie, jego wartość jest zamieniana na 255 (biały) lub 0 (czarny), po wcześniejszej transformacji na obraz monochromatyczny:

$$I'(x, y) = \begin{cases} 255, & \text{jeśli } I(x, y) > T \\ 0, & \text{w przeciwnym razie} \end{cases}$$

gdzie T to ustalony próg binaryzacji.



Rysunek 5: Binary nr 1



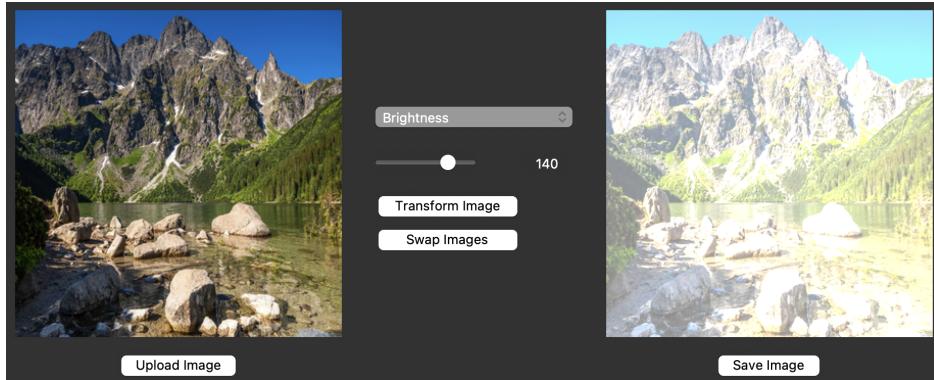
Rysunek 6: Binary nr 2

2.3 Korekcja jasności obrazu

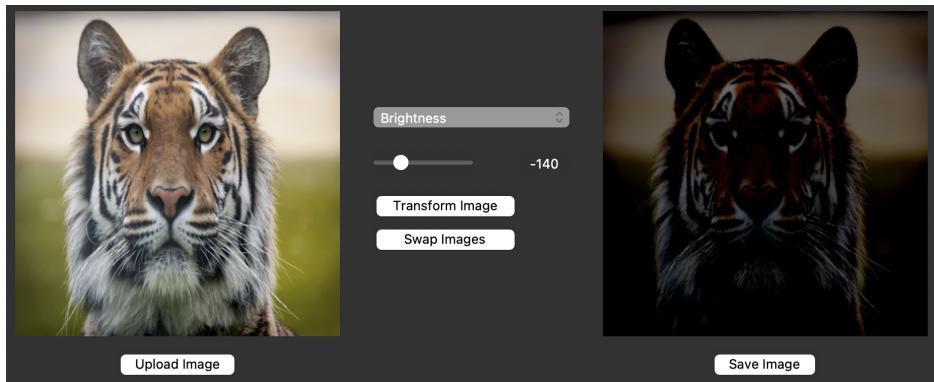
- `def brightness_correction(image, value),`
- `image` - obrazek poddawany modyfikacji jasności,

- **value** - wartość zmiany jasności (może być dodatnia lub ujemna),
- Zastosowanie - zwiększenie lub zmniejszenie jasności obrazu poprzez dodanie wartości do każdego piksela, z zachowaniem ograniczenia do zakresu [0, 255],
- Transformacja - dla każdego piksela $I(x, y)$ w obrazie:

$$I'(x, y) = \text{clip}(I(x, y) + \Delta)$$



Rysunek 7: Brightness nr 1



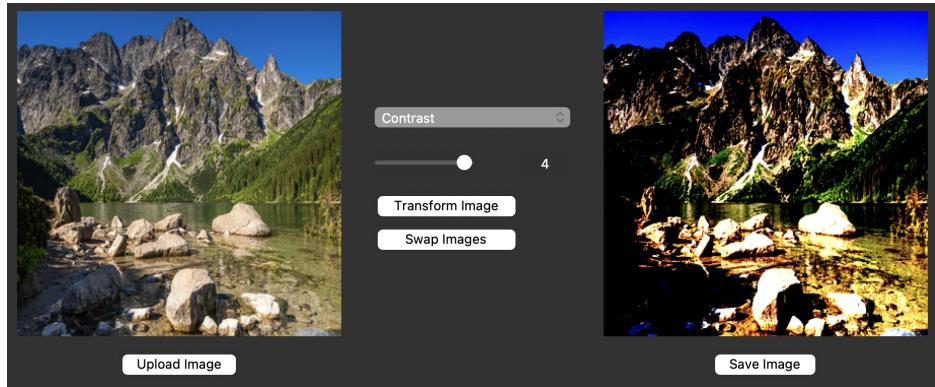
Rysunek 8: Brightness nr 2

2.4 Korekcja kontrastu obrazu

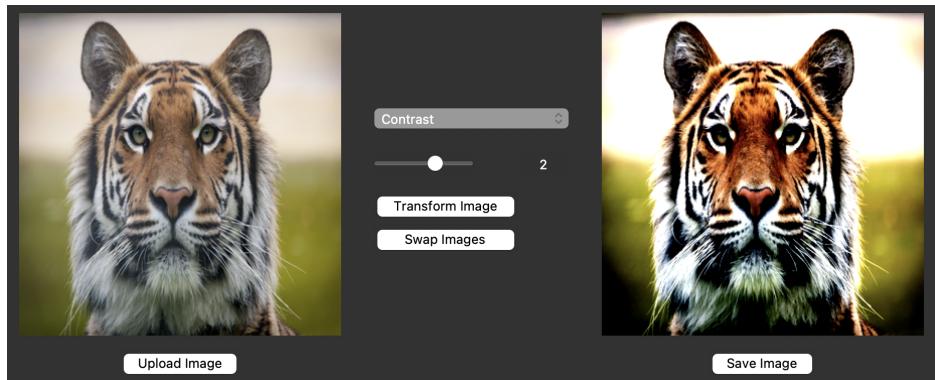
- `def contrast_correction(image, value),`
- **image** - obrazek poddawany modyfikacji kontrastu,
- **value** - współczynnik zmiany kontrastu (wartość dodatnia, zwiększa kontrast),
- Zastosowanie - modyfikacja kontrastu obrazu poprzez zmianę wartości pikseli w stosunku do średniej wartości (128), z zachowaniem ograniczenia do zakresu [0, 255],
- Transformacja - dla każdego piksela $I(x, y)$ w obrazie, jego wartość jest skalowana w zależności od parametru v (współczynnika kontrastu):

$$I'(x, y) = v \cdot (I(x, y) - 128) + 128$$

gdzie v to wartość parametru kontrastu (jeśli $v > 1$, kontrast jest zwiększany; jeśli $v < 1$, kontrast jest zmniejszany).



Rysunek 9: Korekcja kontrastu nr 1

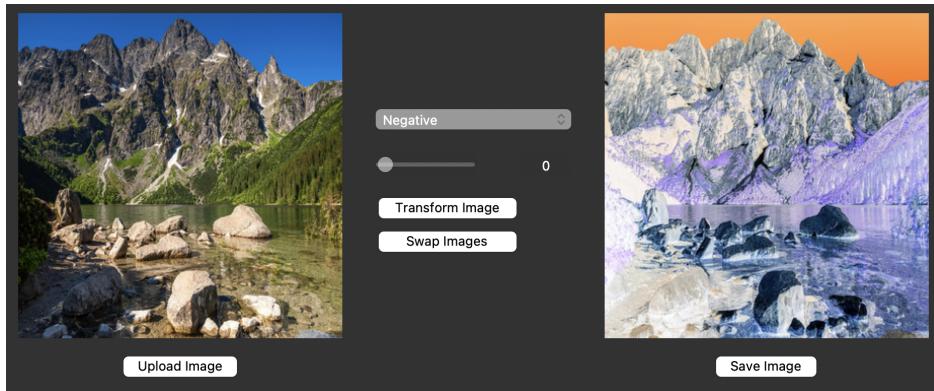


Rysunek 10: Korekcja kontrastu nr 2

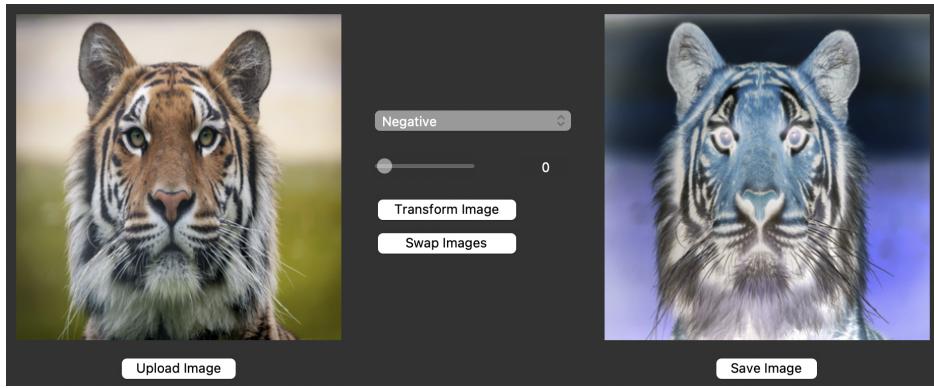
2.5 Filtr negatywu obrazu

- `def negative_filter(image),`
- `image` - obrazek poddawany filtrowaniu,
- Zastosowanie - tworzenie negatywu obrazu przez odwrócenie wartości kolorów pikseli (255 - wartość piksela),
- Transformacja - dla każdego kanału $c \in \{R, G, B\}$ oraz każdego piksela $I_c(x, y)$:

$$I'_c(x, y) = 255 - I_c(x, y)$$



Rysunek 11: Negatyw nr 1



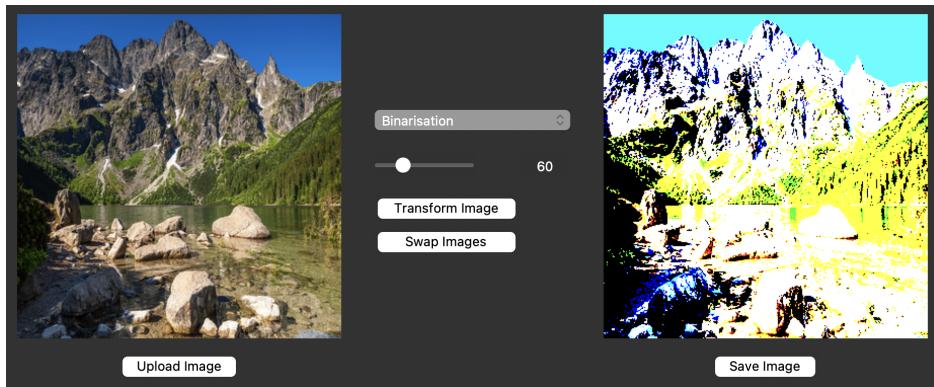
Rysunek 12: Negatyw nr 2

2.6 Binaryzacja obrazu w 3 kolorach

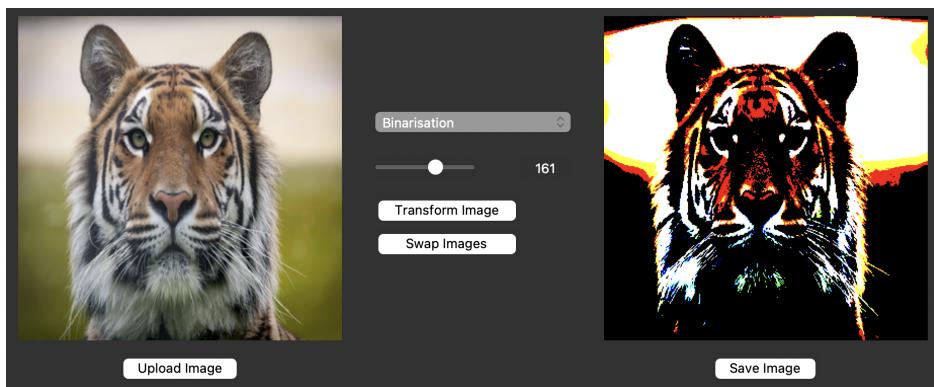
- `def binarisation(image, threshold),`
- `image` - obrazek poddawany filtrowaniu,
- `threshold` - próg, powyżej którego piksele stają się białe (255), poniżej czarne (0),
- Zastosowanie - konwersja obrazu do wersji binarnej, w której piksele są tylko czarne lub białe, w zależności od progu,
- Transformacja - dla każdego kanału $c \in \{R, G, B\}$ oraz każdego piksela $I_c(x, y)$ stosowana jest następująca transformacja progowa:

$$I'_c(x, y) = \begin{cases} 255, & \text{jeśli } I_c(x, y) > T \\ 0, & \text{wpp} \end{cases}$$

gdzie T to ustalony próg binaryzacji.



Rysunek 13: Binaryzacja nr 1

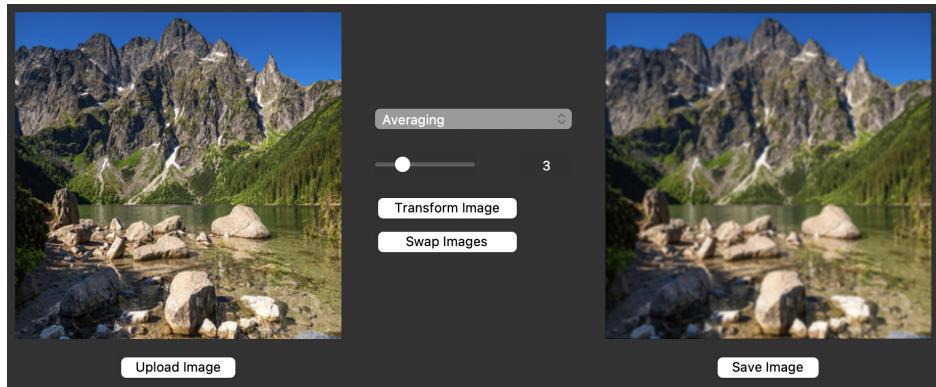


Rysunek 14: Binaryzacja nr 2

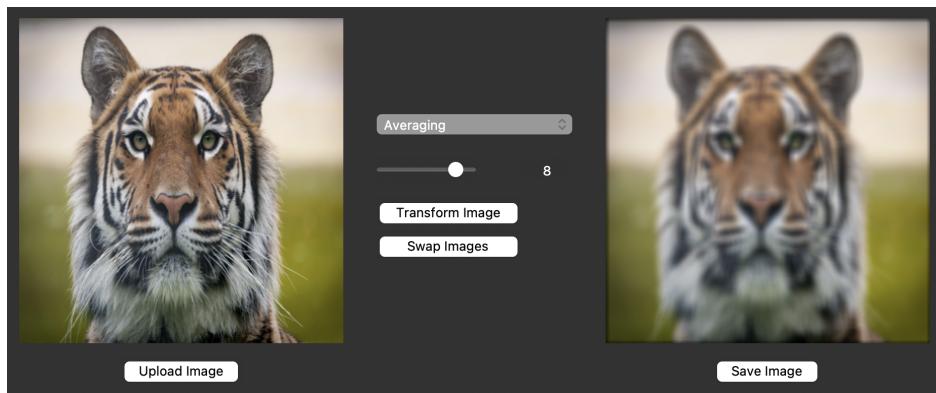
2.7 Filtr uśredniający

- `def averaging(image, kernel_size),`
- `image` - obrazek poddawany filtrowaniu,
- `kernel_size` - rozmiar jądra filtra (np. 3x3, 5x5),
- Zastosowanie - stosowanie filtra uśredniającego w celu wygładzania obrazu, gdzie każdy piksel w obrębie jądra jest zastępowany średnią wartością z sąsiednich pikseli,
- Transformacja -

$$I'(x, y) = \text{mean}\{I(i, j) \mid (i, j) \in \text{kernel}\}$$



Rysunek 15: Filtr uśredniający nr 1

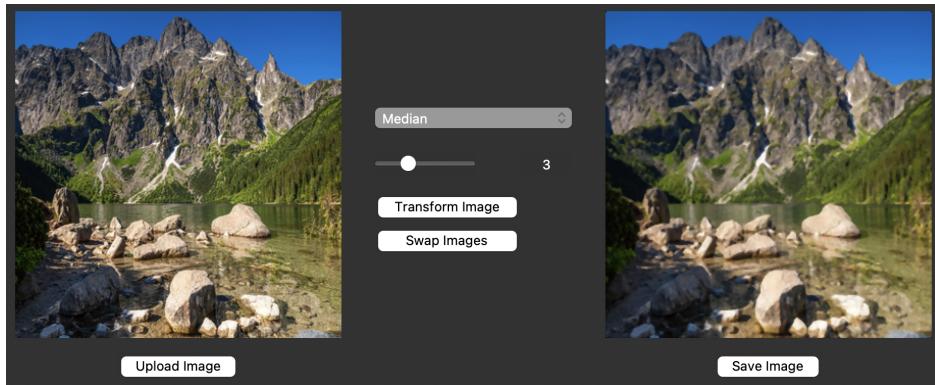


Rysunek 16: Filtr uśredniający nr 2

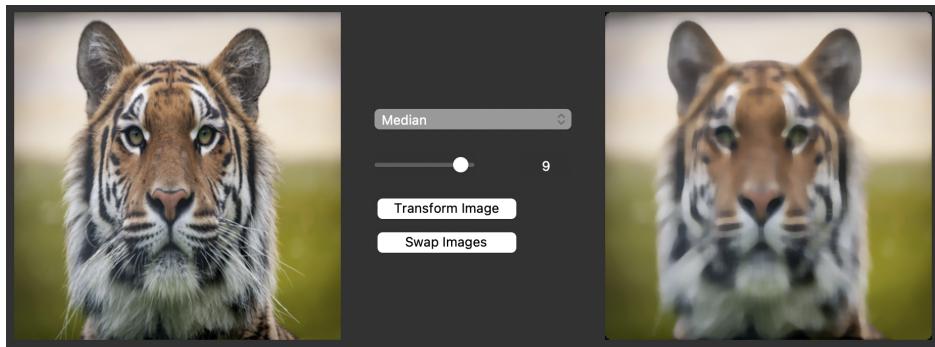
2.8 Filtr medianowy

- `def median(image, kernel_size),`
- `image` - obrazek poddawany filtrowaniu,
- `kernel_size` - rozmiar jądra filtra (np. 3x3, 5x5),
- Zastosowanie - stosowanie filtra medianowego, który zastępuje każdy piksel w obrębie jądra jego medianą, co pomaga w usuwaniu szumów obrazu.
- Transformacja -

$$I'(x, y) = \text{median}\{I(i, j) \mid (i, j) \in \text{kernel}\}$$



Rysunek 17: Filtr medianowy nr 1



Rysunek 18: Filtr medianowy nr 2

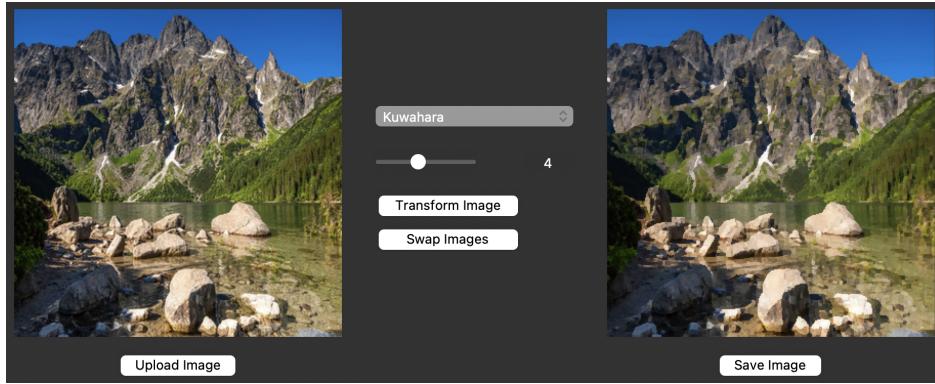
2.9 Filtr Kuwahary

- `def kuwahara(image, kernel_size),`
- `image` - obrazek poddawany filtrowaniu,
- `kernel_size` - rozmiar jądra filtra (np. `3x3, 5x5`),
- Zastosowanie - stosowanie filtra Kuwahary, który jest techniką wygładzania obrazu, w której dla każdego piksela obliczana jest średnia z najmniej zmennego regionu w obrębie jądra. Filtr ten jest szczególnie użyteczny w usuwaniu szumów, zachowując krawędzie,
- Transformacja - dla każdego piksela obliczane są średnie wartości jasności i wariancje w czterech sąsiadujących oknach:

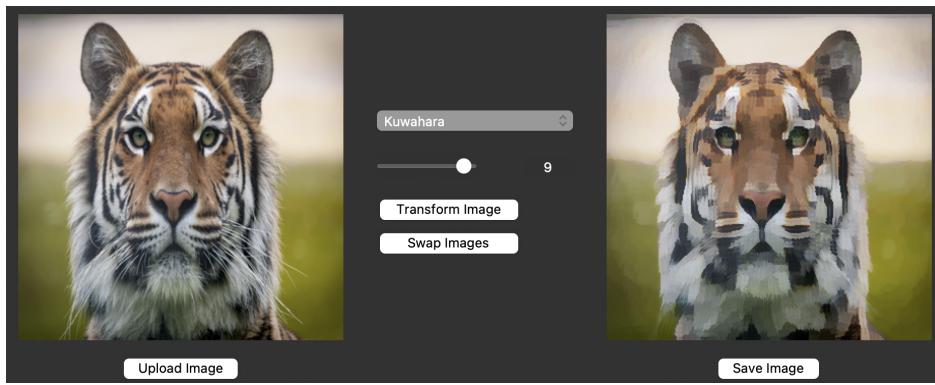
$$\mu_k = \frac{1}{N} \sum_{(i,j) \in \text{subobszar}_k} I(i,j)$$

$$\sigma_k^2 = \frac{1}{N} \sum_{(i,j) \in \text{subobszar}_k} (I(i,j) - \mu_k)^2$$

gdzie $I(i,j)$ to wartość piksela, μ_k - średnia jasność, a σ_k^2 - wariancja dla danego podobszaru k .



Rysunek 19: Filtr Kuwahary nr 1



Rysunek 20: Filtr Kuwahary nr 2

2.10 Filtr Gaussa

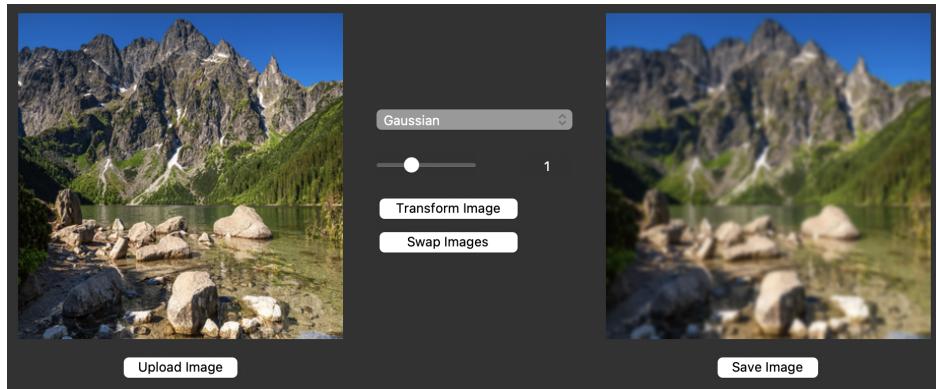
- def gaussian(image, sigma),
- image - obrazek poddawany filtrowaniu,
- sigma - odchylenie standardowe rozkładu Gaussa, które wpływa na rozmycie obrazu (im większe, tym mocniejsze rozmycie),
- Zastosowanie - stosowanie filtra Gaussa, który wygładza obraz poprzez stosowanie rozkładu Gaussa w celu zminimalizowania szumów i detali,
- Transformacja - zastosowanie splotu obrazu z macierzą opartą na rozkładzie Gaussa:

$$G = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

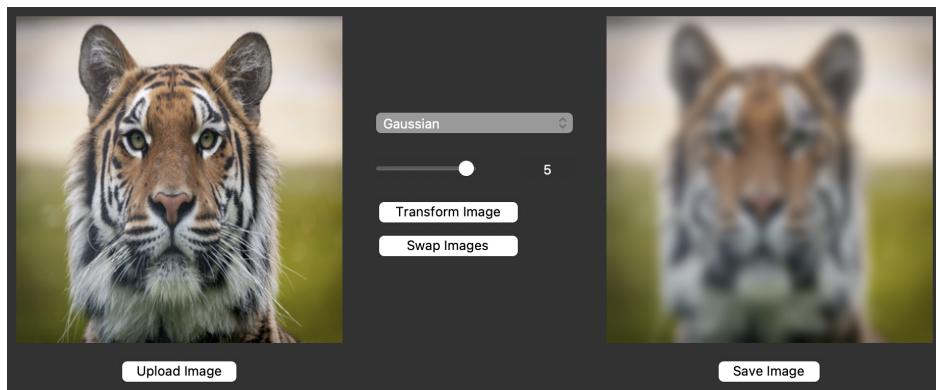
Ogólna postać macierzy dla rozmiaru $(2k + 1) \times (2k + 1)$ i odchylenia standardowego σ :

$$G(i, j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2 + j^2}{2\sigma^2}\right)$$

gdzie i, j to współrzędne względem środka macierzy filtru.



Rysunek 21: Filtr Gaussa nr 1



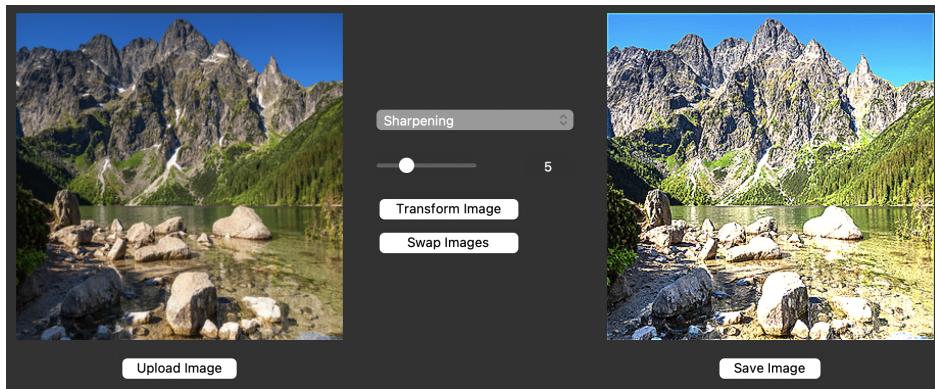
Rysunek 22: Filtr Gaussa nr 2

2.11 Filtr wyostrzający

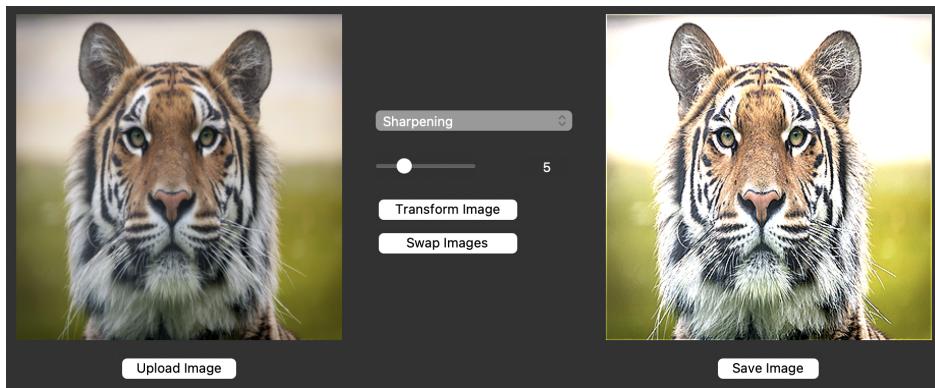
- `def sharpening(image, value),`
- `image` - obrazek poddawany filtrowaniu,
- `value` - współczynnik wyostrzania (im wyższa wartość, tym mocniejsze wyostrzenie),
- Zastosowanie - stosowanie filtra wyostrzającego w celu zwiększenia kontrastu krawędzi w obrazie, poprzez wzmacnienie różnic między sąsiednimi pikselami,
-
- Transformacja - przykładowa macierz konwolucyjna podkreślająca zmiany jasności:

$$G = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Wartość centralna określa stopień wyostrzenia.



Rysunek 23: Filtr wyostrzajacy nr 1



Rysunek 24: Filtr wyostrzajacy nr 2

2.12 Operator Robertsa

- `def roberts(image),`
- `image` - obrazek poddawany filtrowaniu,
- Zastosowanie - stosowanie operatora Robertsa w celu wykrywania krawędzi w obrazie. Jest to operator różnicowy, który oblicza różnice między pikselami w kierunku pionowym i poziomym,
- Transformacja - obliczanie gradientu obrazu za pomocą skośnych macierzy konwolucyjnych:

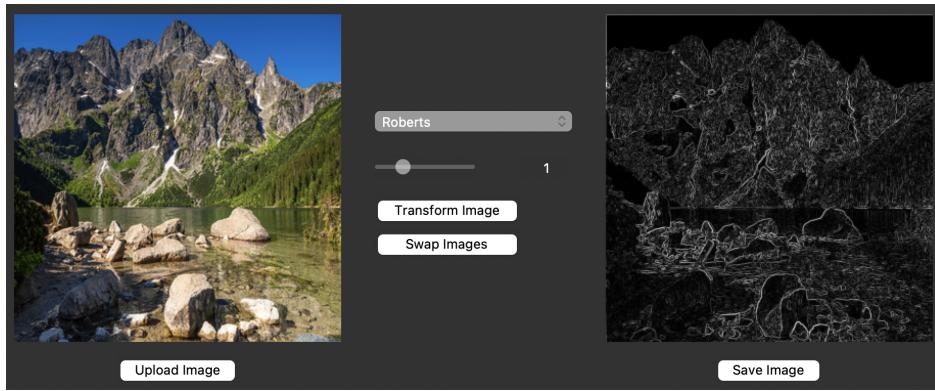
$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Gradient końcowy wyznaczany jest jako:

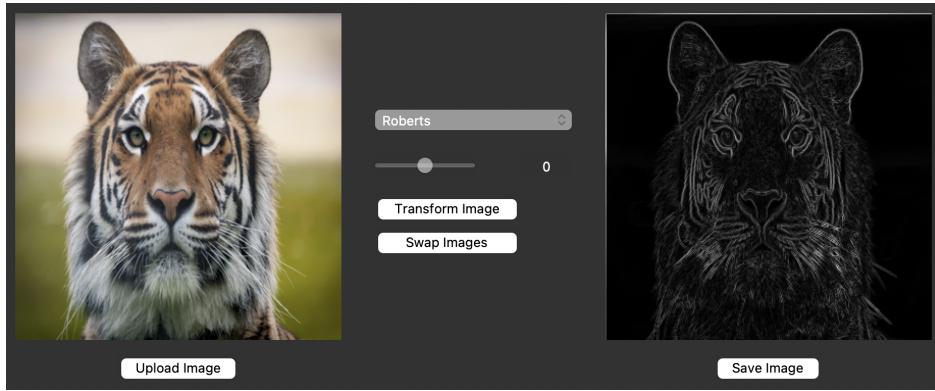
$$G = \sqrt{G_x^2 + G_y^2}$$

lub w przybliżeniu:

$$G \approx |G_x| + |G_y|$$



Rysunek 25: Wykrywanie krawędzi - operator Roberts nr 1



Rysunek 26: Wykrywanie krawędzi - operator Roberts nr 2

2.13 Operator Sobela

- `def sobel(image),`
- `image` - obrazek poddawany filtrowaniu,
- Zastosowanie - stosowanie operatora Sobela w celu wykrywania krawędzi w obrazie. Operator ten oblicza gradient obrazu w kierunkach poziomym i pionowym, co pozwala na wydobycie krawędzi,
- Transformacja - Operator Sobela oblicza gradient obrazu, wzmacniając krawędzie poprzez splot z macierzami w kierunkach poziomym i pionowym:

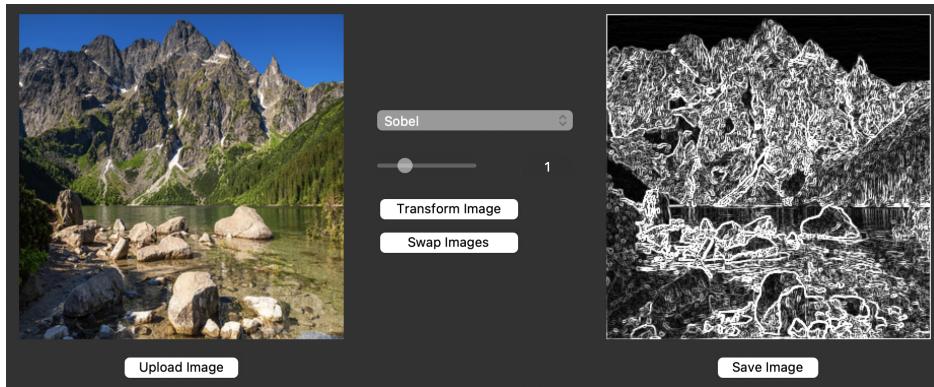
$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Gradient końcowy wyznaczany jest jako:

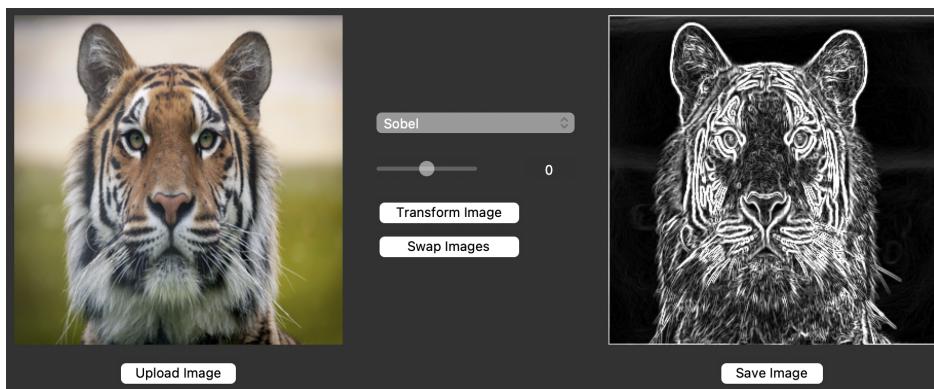
$$G = \sqrt{G_x^2 + G_y^2}$$

lub w przybliżeniu:

$$G \approx |G_x| + |G_y|$$



Rysunek 27: Wykrywanie krawędzi - operator Sobela nr 1



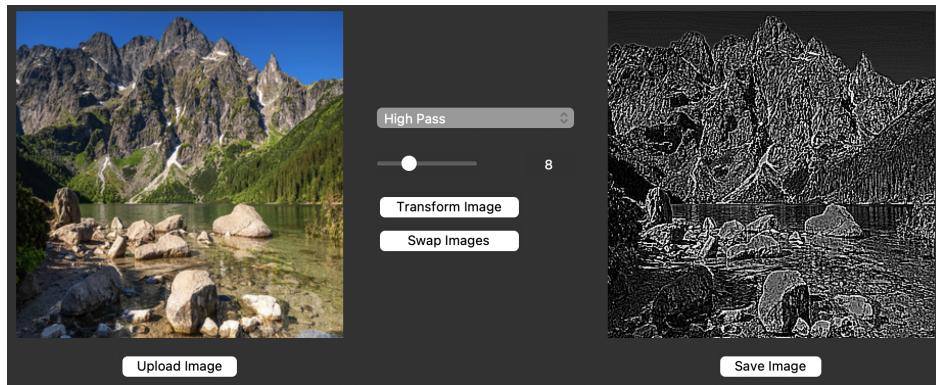
Rysunek 28: Wykrywanie krawędzi - operator Sobela nr 2

2.14 Filtr górnoprzepustowy

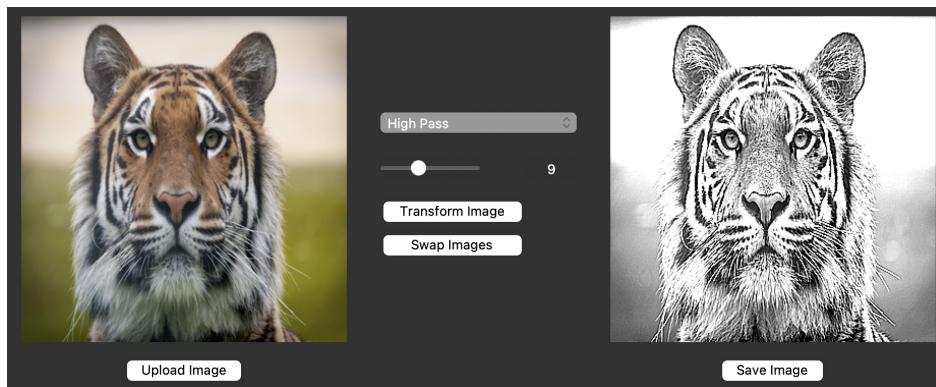
- `def high_pass(image, value),`
- `image` - obrazek poddawany filtrowaniu,
- `value` - wartość, która wpływa na intensywność filtrowania (im wyższa wartość, tym bardziej wyeksponowane są wysokie częstotliwości obrazu),
- Zastosowanie - stosowanie filtra górnoprzepustowego w celu uwydatnienia detali i krawędzi obrazu, poprzez usunięcie komponentów niskoczęstotliwościowych (np. tła lub większych struktur),
- Transformacja - przykładowa macierz filtru górnoprzepustowego:

$$G = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Wartość środkowa może być dostosowana.



Rysunek 29: Filtr górnoprzepustowy nr 1



Rysunek 30: Filtr górnoprzepustowy nr 2

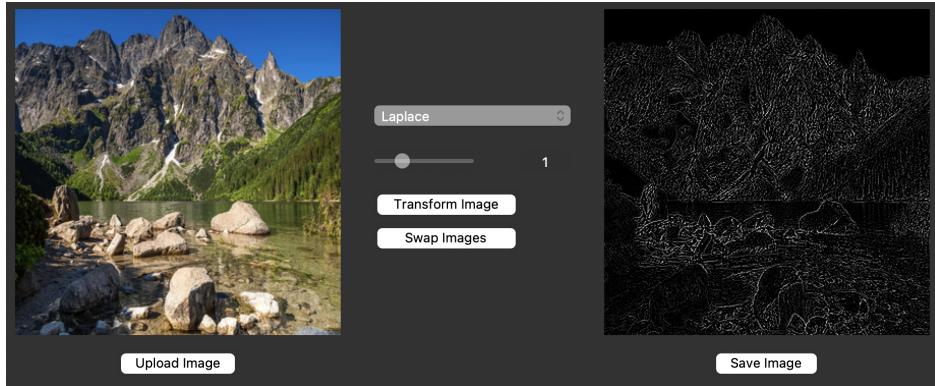
2.15 Operator Laplace'a

- `def laplace(image),`
- `image` - obrazek poddawany filtrowaniu,
- Zastosowanie - stosowanie operatora Laplace'a do wykrywania krawędzi w obrazie. Operator Laplace'a jest drugim różniczkowym operatorem, który oblicza drugą pochodną obrazu, co umożliwia wykrywanie punktów, w których zmienia się intensywność (np. krawędzi).
- Transformacja - macierz filtru Laplace'a:

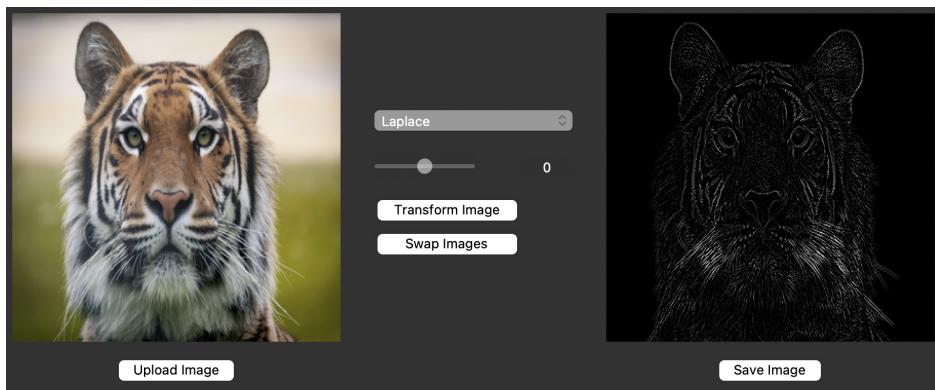
$$G = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Alternatywna wersja:

$$G = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



Rysunek 31: Wykrywanie krawędzi - operator Laplace'a nr 1



Rysunek 32: Wykrywanie krawędzi - operator Laplace'a nr 2

2.16 Operator Prewitta

- `def prewitt(image),`
- `image` - obrazek poddawany filtrowaniu,
- Zastosowanie - stosowanie operatora Prewitta w celu wykrywania krawędzi w obrazie. Operator ten oblicza gradient obrazu w kierunkach poziomym i pionowym, co pozwala na wyodrębnienie krawędzi, podobnie jak operator Sobela, ale używając innego jądra,
- Transformacja - gradient obrazu obliczany jest jako:

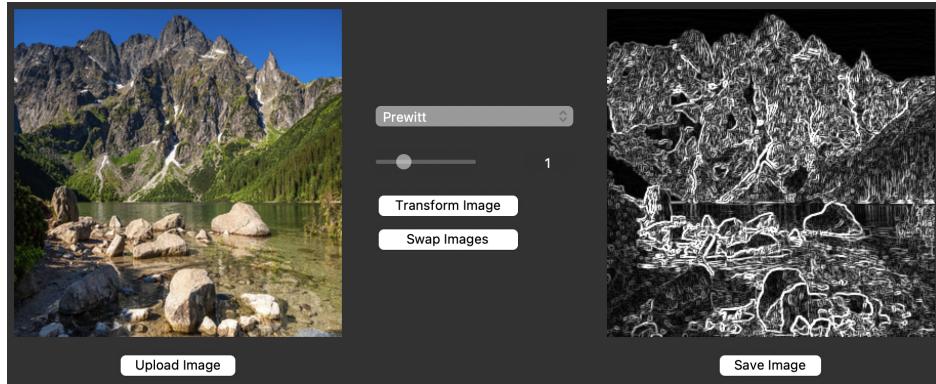
$$G = \sqrt{G_x^2 + G_y^2}$$

lub w przybliżeniu:

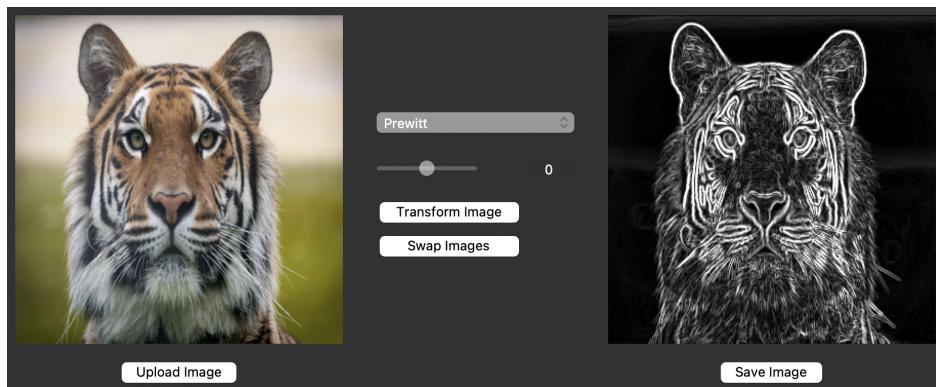
$$G \approx |G_x| + |G_y|$$

Macierze dla kierunków poziomego i pionowego:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$



Rysunek 33: Wykrywanie krawędzi - operator Prewitta nr 1



Rysunek 34: Wykrywanie krawędzi - operator Prewitta nr 2

2.17 Filtr Ridge

- `def ridge(image),`
- `image` - obrazek poddawany filtrowaniu,
- Zastosowanie - filtr ridge jest stosowany do wykrywania strukturalnych cech obrazu, takich jak krawędzie lub wypukłości. Często wykorzystywany w analizie tekstur i rozpoznawaniu wzorców, szczególnie w kontekście obrazów medycznych i geologicznych,
- Transformacja - gradient obrazu obliczany jest jako:

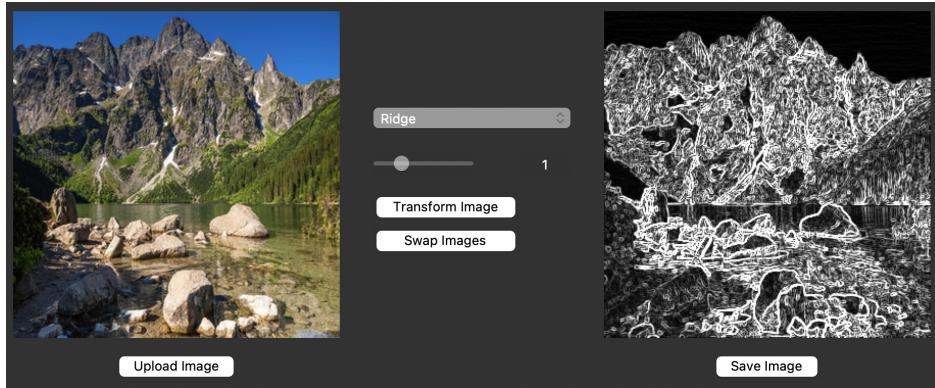
$$G = \sqrt{G_x^2 + G_y^2}$$

lub w przybliżeniu:

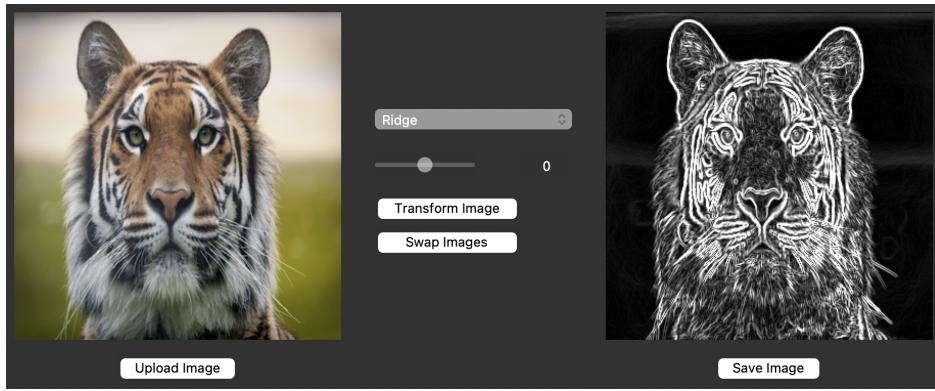
$$G \approx |G_x| + |G_y|$$

Macierze dla kierunków poziomego i pionowego:

$$G_x = \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}, \quad G_y = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$



Rysunek 35: Filtr Ridge nr 1



Rysunek 36: Filtr Ridge nr 2

2.18 Operator Scharra

- `def scharr(image),`
- `image` - obrazek poddawany filtrowaniu,
- Zastosowanie - stosowanie operatora Scharra w celu wykrywania krawędzi w obrazie. Jest to modyfikacja operatora Sobela, która stosuje inne wagę dla obliczania gradientów, co daje lepsze wyniki w detekcji krawędzi w niektórych przypadkach.
- Transformacja - gradient obrazu obliczany jest jako:

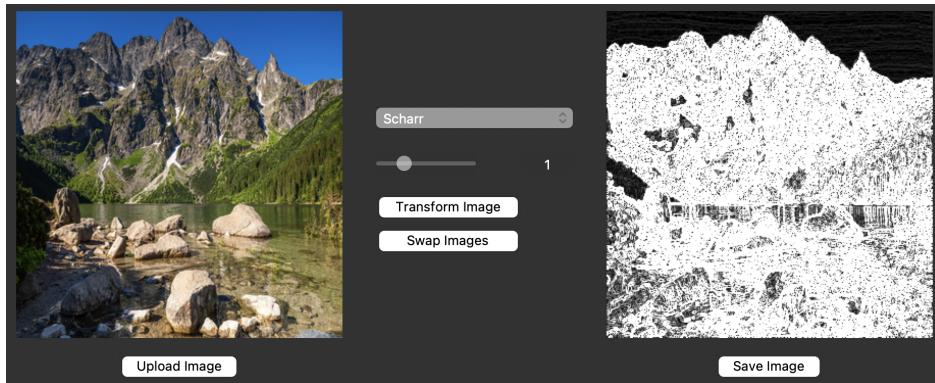
$$G = \sqrt{G_x^2 + G_y^2}$$

lub przybliżeniem:

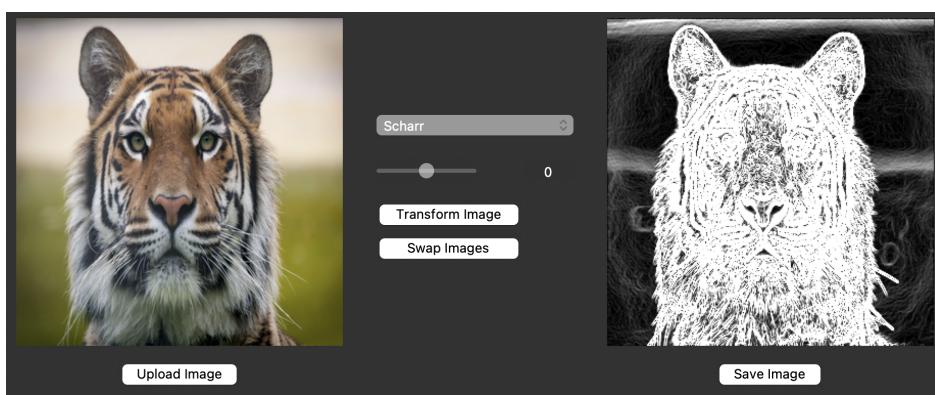
$$G \approx |G_x| + |G_y|$$

Macierze dla kierunków poziomego i pionowego:

$$G_x = \begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}, \quad G_y = \begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}$$



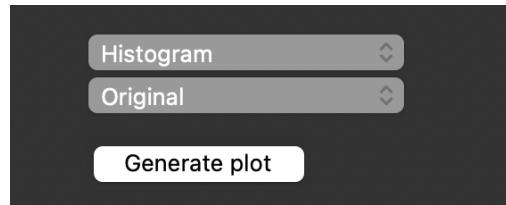
Rysunek 37: Wykrywanie krawędzi - operator Scharra nr 1



Rysunek 38: Wykrywanie krawędzi - operator Scharra nr 2

3 Wykresy

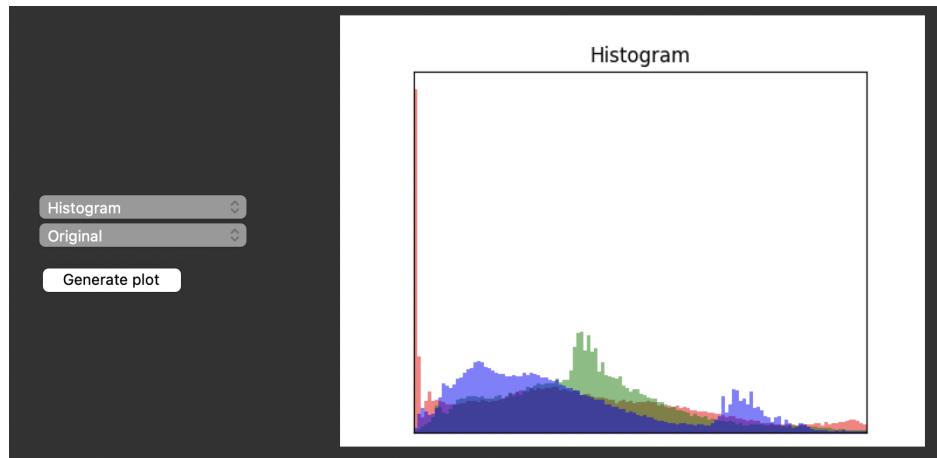
Istnieje możliwość wygenerowania jednego z trzech wykresów w celu analizy obrazu, zarówno wejściowego jak i obrazu po transformacji.



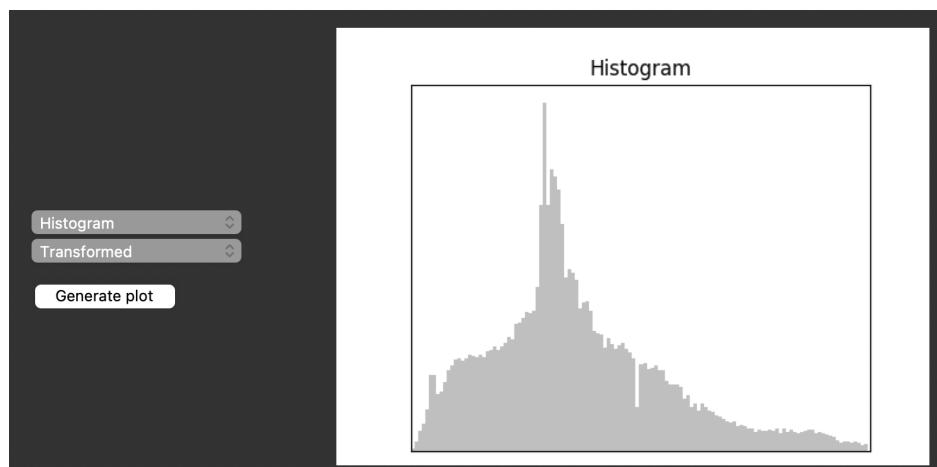
Rysunek 39: Opcje wykresu

3.1 Histogram

Zliczamy występowanie każdego piksela danej barwy oraz przedstawienie w postaci histogramu. Dostępne również dla obrazów monochromatycznych.



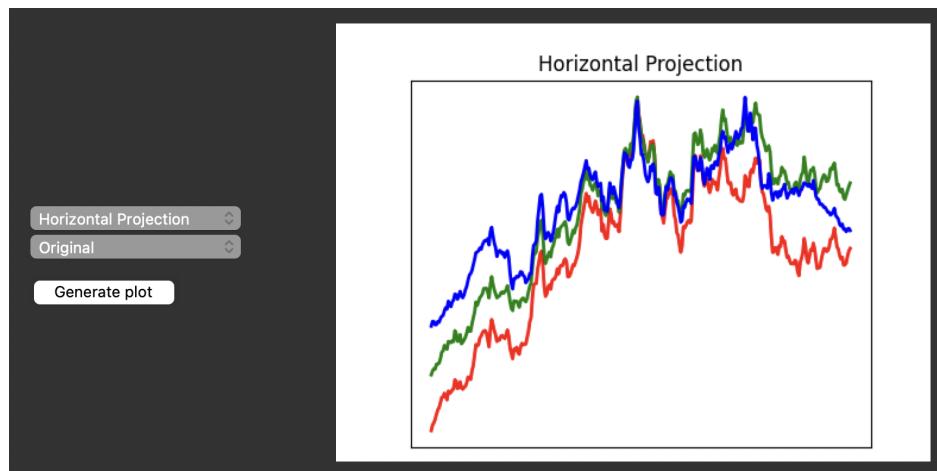
Rysunek 40: Histogram dla kolorowego obrazka



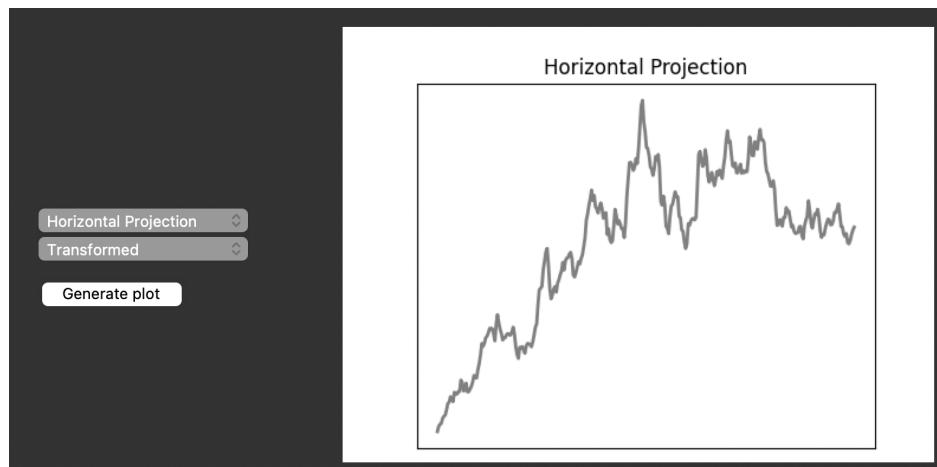
Rysunek 41: Histogram dla szarego obrazka

3.2 Projekcja Pozioma

Suma pikseli w każdej kolumnie z podziałem na kanały.



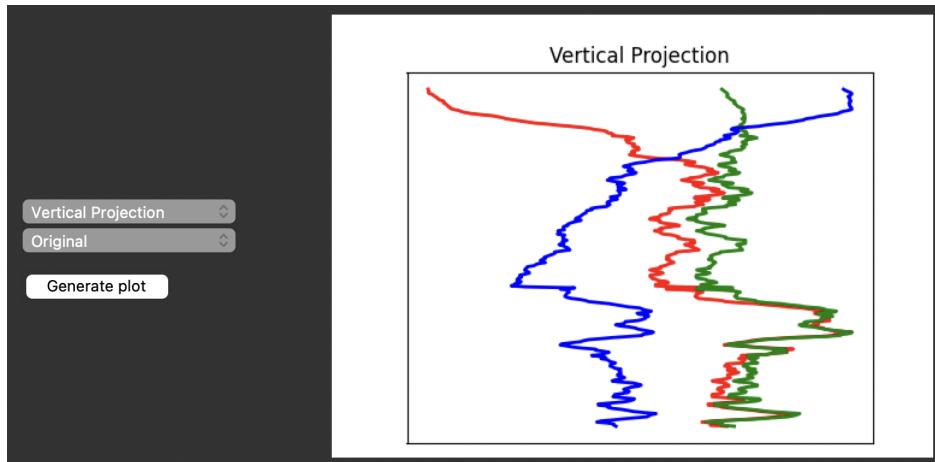
Rysunek 42: Projekcja pozioma dla kolorowego obrazka



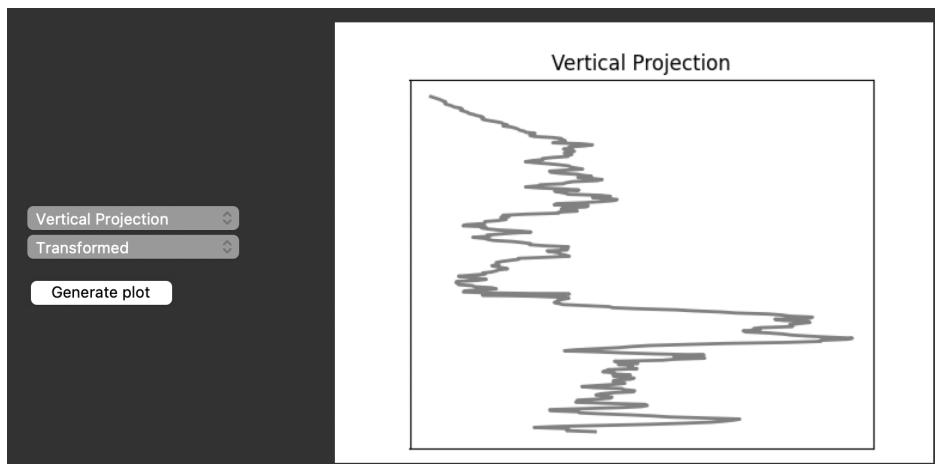
Rysunek 43: Projekcja pozioma dla szarego obrazka

3.3 Projekcja Pionowa

Suma pikseli w każdym wierszu.



Rysunek 44: Projekcja pozioma dla pionowa obrazka

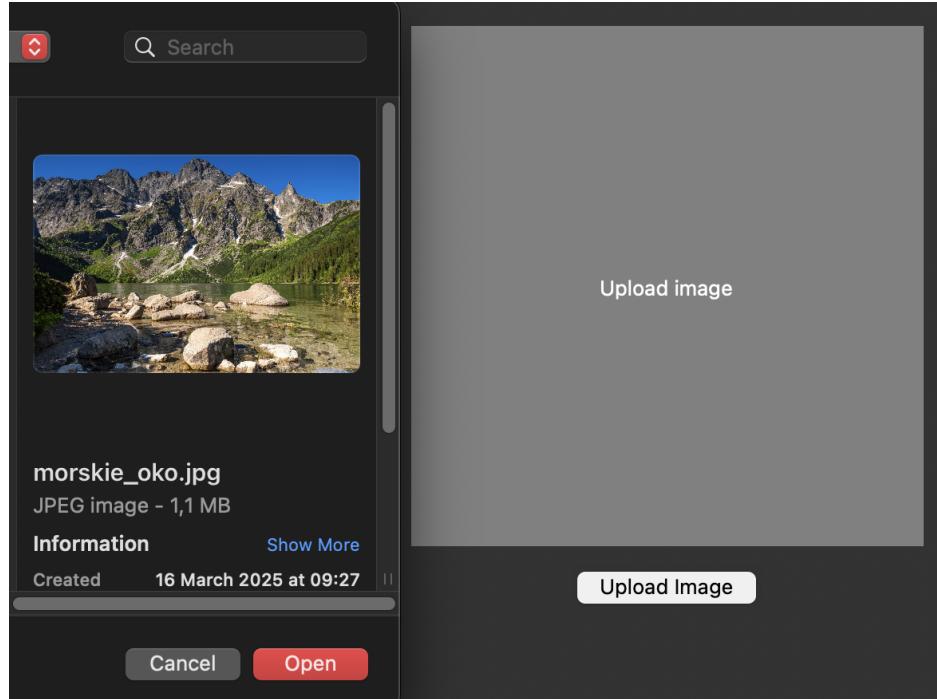


Rysunek 45: Projekcja pionowa dla szarego obrazka

4 Inne funkcjonalności

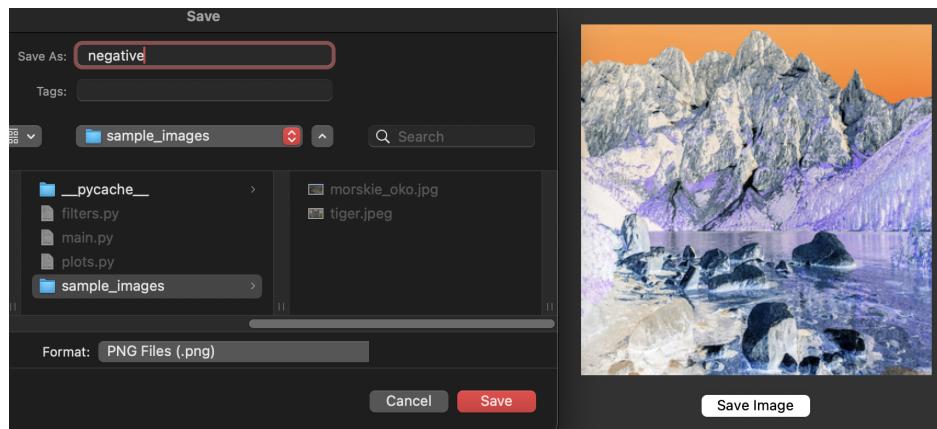
Dodatkowe opcje, które zostały zaimplementowane:

- Wczytanie obrazku z plików urządzenia (pliki png i jpeg),



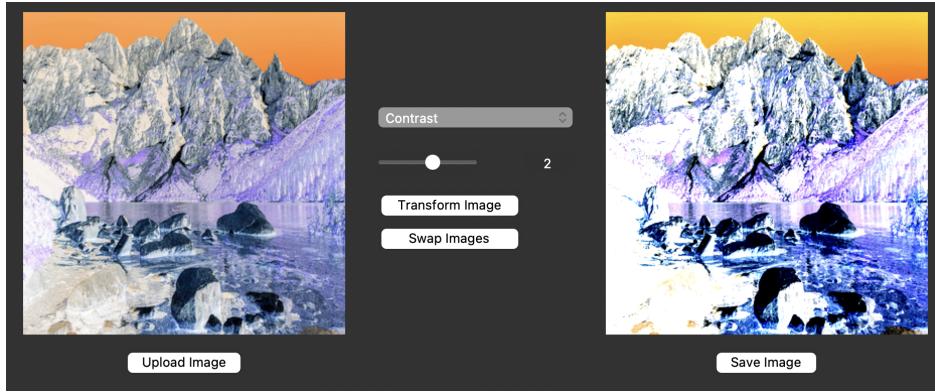
Rysunek 46: Wczytywanie pliku

- Zapis obrazku na dysk użytkownika,



Rysunek 47: Zapis obrazu

- Dalsza edycja obrazka po transformacji.



Rysunek 48: Wykorzystanie opcji "Swap Images" w celu użycia negatywu oraz poprawy kontrastu

5 Podsumowanie

Stworzona aplikacja umożliwia przetwarzanie obrazów poprzez zastosowanie różnorodnych filtrów i operacji na pikselach. Użytkownik może wczytywać oraz zapisywać obrazy, a także nakładać na nie wybrane efekty, takie jak konwersja do skali szarości, korekta jasności i kontrastu, negatyw czy binaryzacja.

Dostępne są również bardziej zaawansowane filtry, m.in. wygładzające (uśredniający, Gaussa, Kuwahary), wykrywające krawędzie (Sobel, Scharr, Prewitt, Laplace, Roberts) oraz wyostrzające. Aplikacja oferuje histogram obrazu oraz projekcje pionową i poziomą.

Interfejs użytkownika pozwala na intuicyjne wybieranie filtrów za pomocą rozwijanego menu oraz dostosowywanie ich parametrów przy użyciu suwaka. Implementacja wykorzystuje biblioteki Tkinter do obsługi GUI, NumPy do operacji na macierzach oraz Pillow do przetwarzania obrazów.