Dokumentacja Aplikacji - Projekt 1

Bartosz Jezierski

Spis treści

1 Wprowadzenie

Aplikacja **Image Editor** to narzędzie do przetwarzania obrazów, stworzone jako projekt edukacyjny.

Główna Klasa:

`ImageEditor` – odpowiada za inicjalizację interfejsu, obsługę zdarzeń (kliknięcia, zmiany suwaków) oraz realizację operacji przetwarzania obrazu.

Biblioteki:

- **Tkinter:** Budowa interfejsu graficznego.
- **Pillow (PIL):** Otwieranie, zapisywanie i konwersja obrazów.
- **NumPy:** Wykonywanie operacji matematycznych i przetwarzania obrazu w postaci tablic.
- **Matplotlib:** Generowanie wykresów histogramów oraz projekcji.

2 Wymagania Projektowe

Na ocenę 3.0

- Aplikacja okienkowa (GUI) w dowolnym języku.
- Operacje na pikselach:
  - Konwersja do odcieni szarości,
  - Korekta jasności,
  - Korekta kontrastu,
  - Negatyw,
  - Binaryzacja.
- Filtry graficzne:
  - Uśredniający,
  - Gaussa,
  - Wyostrzający.
- Dokumentacja.

Na ocenę 4.0 (rozszerzenie funkcjonalności)

- Opcja zapisu pliku przetworzonego.
- Histogram.
- Projekcje pozioma i pionowa.
- Wykrywanie krawędzi:
  - Krzyż Robertsa,
  - Operator Sobela,
  - Operator Prewitta.
- Możliwość tworzenia dowolnych wag filtrów (własny filtr konwolucyjny).

3 Opis Funkcjonalności

3.1 Konwolucja

Operacja zastosowania filtra jest zaimplementowana w funkcjach `convolution_gray` i `convolution`. Z tych funkcji korzystają wszystkie funkcjonalności z sekcji 3.3 oraz 3.4. Zanim zostanie zastosowany filtr zdjęcie rozszerzane jest pixelami brzegowymi, tak, żeby zdjęcie nie zmieniło swoich początkowych wymiarów. Poniżej zamieszczam grafikę tłumaczącą logikę wypełnienia brzegów.

Rysunek 1: Wizualizacja wypełnienia brzegów

W celu zrozumienia operacji konwolucji odsyłam do wikipedii. Operacja konwolucji zaimplementowana została w pętli python'owej na prośbę prowadzącego przedmiot. Z tego powodu wykonuje się bardzo długo. Aby przyspieszyć tę operację wystarczy wyrzucić pętle i skorzystać z funkcji `np.lib.stride_tricks.sliding_window_view()`.

3.2 Operacje na Pikselach

Konwersja do Odcieni Szarości:
Funkcja `to_grayscale` przetwarza obraz kolorowy, zamieniając go na skalę szarości poprzez zastosowanie dla każdego pixela wzoru:

$$new\_pixel = 0.2989r + 0.5870g + 0.1140b$$

Negatyw:
Funkcja `to_negative` tworzy negatyw obrazu poprzez odwrócenie wartości pikseli (obliczając 255 − wartość_piksela).

Binaryzacja:
Funkcja `binarize` przekształca obraz w odcieniach szarości na postać binarną (czarno-białą) na podstawie zadanego progu. Jeżeli podany jest obraz z 3 kanałami to najpierw jest on konwertowany do odcieni szarości i następnie stosowany jest próg według poniższego wzoru. Użytkownik ma możliwość wprowadzenia progu w oknie dialogowym.

$$f(x) = \begin{cases} 0, & \text{dla } x < threshold, \\ 255, & \text{dla } x \geq threshold. \end{cases}$$

Korekta Jasności i Kontrastu:
Funkcja `apply_contrast_and_brightness` korzysta z suwaków, umożliwiając regulację jasności oraz kontrastu obrazu. Wykorzystywany jest współczynnik kontrastu oparty o wzór:

$$\text{factor} = \frac{259 \times (\text{contrast} + 255)}{255 \times (259 - \text{contrast})}$$

następnie każdy piksel jest przeliczany według wzoru:

$$\text{new\_pixel} = \text{factor} \times (\text{old\_pixel} - 128) + 128 + \text{brightness}$$

Wynik jest ograniczany do przedziału 0–255.

3.3 Filtry Graficzne

Average Blur (Filtr Uśredniający):
Implementowany poprzez funkcję `filter_average`, która wykorzystuje jądro (macierz) o wymiarach 5×5 z równymi wagami $\frac{1}{25}$.

Gaussian Blur (Filtr Gaussa):
Funkcja `filter_gaussian` stosuje jądro Gaussa:

$$\text{kernel} = \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} : 273.0$$

Sharpen (Wyostrzanie):
Funkcja `filter_sharpen` korzysta z klasycznego jądra wyostrzającego, które uwydatnia krawędzie i detale obrazu:

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

Custom Filter (Filtr Niestandardowy):
Zapewione przez funckję `filter_custom`. Użytkownik może zdefiniować własne jądro filtrujące, wprowadzając wagi w interaktywnym oknie dialogowym.

3.4 Wykrywanie Krawędzi

Rysunek 2: Przykładowe użycie wykrywania krawędzi filtrami Sobela

Operator Roberts:
Funkcja `filter_roberts` wykorzystuje dwa jądra do wykrywania krawędzi metodą Roberta.

$$\text{kernel1} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$\text{kernel2} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Dwa obrazy wynikowe są łączone przy użyciu pierwiastka sumy kwadratów po pixelach:

$$pixel\_new = \sqrt{pixel^2_{kernel1} + pixel^2_{kernel2}}$$

Operator Sobela:
Funkcja `filter_sobel` stosuje dwa jądra do wykrywania krawędzi w kierunkach poziomym i pionowym:

$$\text{kernel1} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\text{kernel2} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Dwa obrazy wynikowe są łączone przy użyciu pierwiastka sumy kwadratów po pixelach:

$$pixel\_new = \sqrt{pixel^2_{kernel1} + pixel^2_{kernel2}}$$

Operator Prewitta:
Analogicznie do Sobela, funkcja `filter_prewitt` wykorzystuje jądra Prewitta do wykrywania krawędzi.

$$\text{kernel1} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\text{kernel2} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Dwa obrazy wynikowe są łączone przy użyciu pierwiastka sumy kwadratów po pixelach:

$$pixel\_new = \sqrt{pixel^2_{kernel1} + pixel^2_{kernel2}}$$

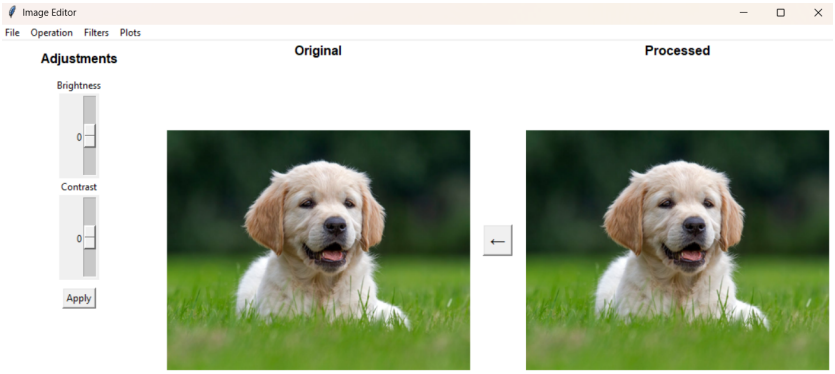
3.5 Dodatkowe Funkcje

Zapisywanie Obrazu:
Opcja zapisu przetworzonego obrazu do pliku (PNG, JPEG) dostępna jest poprzez menu `File > Save Image`. Zapisywany jest obraz przetworzony.

Histogram:
Funkcja `show_histogram` generuje histogram obrazu. W przypadku obrazów kolorowych histogramy są tworzone osobno dla kanałów: szarości, czerwonego, zielonego i niebieskiego.

Projekcje Pozioma i Pionowa:
Funkcje `show_projections_vertical` oraz `show_projections_horizontal` prezentują wykresy sum intensywności pikseli w rzędach (projekcja pionowa) oraz kolumnach (projekcja pozioma). Dla obrazów kolorowych wykresy generowane są osobno dla poszczególnych kanałów.

4 Interfejs Użytkownika



Rysunek 3: Interfejs użytkownika

Aplikacja dzieli główne okno na trzy sekcje:

- **Panel Lewy:** Zawiera suwak do regulacji jasności (od -100 do 100) oraz kontrastu (od -128 do 128), a także przycisk "Apply", który stosuje zmiany do obrazu.
- **Panel Środkowy:** Wyświetla oryginalny obraz. Użytkownik może wczytać obraz przy pomocy opcji `File > Open Image`.

- **Panel Prawy:** Pokazuje przetworzony obraz. Dodatkowo, znajduje się przycisk (`<-`) umożliwiający przeniesienie przetworzonego obrazu do panelu oryginalnego (gdyż wszystkie operacje przetwarzające obraz są stosowane na obrazie orginalnym) (funkcja `processed_to_original`).

Menu aplikacji zawiera następujące sekcje:

- **File:** Otwieranie i zapisywanie obrazów.
- **Operation:** Operacje na pikselach (konwersja do szarości, negatyw, binaryzacja).
- **Filters:** Zestaw filtrów graficznych (uśredniający, Gaussa, wyostrzający, wykrywanie krawędzi, filtr niestandardowy).
- **Plots:** Wizualizacja histogramów oraz projekcji poziomych i pionowych.

5 Instrukcja Uruchomienia

Wymagania Systemowe:

- Python (wersja 3.x)
- Biblioteki: Tkinter, Pillow, NumPy, Matplotlib

Instalacja Wymaganych Bibliotek:

Można zainstalować wymagane pakiety za pomocą menedżera `pip`, np.:

```
pip install pillow numpy matplotlib tk
```

Uruchomienie Aplikacji:

1. Uruchom skrypt Pythona zawierający kod aplikacji (`main.py`).
2. Po uruchomieniu, domyślnie ładowany jest przykładowy obraz (np. `test/imgs/dog.jpg`).
3. Aby wczytać inny obraz, skorzystaj z opcji `File > Open Image`.
4. Po wykonaniu operacji przetwarzania obrazu, wynikowy obraz można zapisać przy użyciu opcji `File > Save Image`.

6 Wnioski

- Metoda konwolucji jest za długa przez to, że jest zrobiona w zwykłym python'ie. Jeżeli projekt będzie miał być potem używany przerobię tę funkcję tak, żeby korzystała tylko z funkcji z pakietu `numpy`.
- Podczas zapisywania i filtracji pojawiały się problemy zapisywaniem i przetwarzaniem obrazu. Zakładam, że ma to związek z tym, że podczas różnych operacji zdjęcie "w międzyczasie"jest przetwarzane z wartościami po przecinku (próbowałem to w różnych miejscach naprawiać, ale i tak jest z tym problem).
- Jeżeli stosujemy powiedzmy funkcjonalność `average_blur` badź dowolny inny filtr to jego wpływ będzie znacznie mniej widoczny na zdjęciach o wyższej rozdzielczości. Można zastanowić się nad dostosowaniem wielkości filtra do rozdzielczości zdjęcia.