Przetwarzanie Obrazów i Sygnałów Audio
Projekt

Rok akademicki 2019/2020
WARiE, AIR, SW, sem. 1

Temat: Aplikacja edukacyjna przedstawiająca podstawowe operacje na obrazach

Wykonali:
Marcin Nawrocki (132103)
Tomasz Jankowski (132062)

Spis treści

1 Wstęp

1.1 Założenia projektowe

Projekt zakłada opracowanie aplikacji okienkowej do demonstracji podstawowych przekształceń obrazów możliwej do uruchomienia na platformach z systemem Windows (zarówno dla architektury 32-, jak i 64-bitowej). Aplikacja pozwala na wybór dowolnego obrazu zapisanego w formatach *.jpg*, *.jpeg*, *.png* oraz przeprowadzenie na nim jednej z dostępnych 13 zróżnicowanych operacji. Ich lista zostanie przedstawiona jest poniżej. Po wykonanej operacji można dowolnie przeglądać wszystkie wygenerowane obrazy - od początkowego, przez pośrednie, do wynikowego, a także parametry oraz histogramy obrazów początkowego oraz końcowego. Aplikacja bazuje na samodzielnie opracowanej bibliotece w języku *Python*, natomiast obsługa interfejsu graficznego oraz uruchamiania poszczególnych operacji została napisana w języku *JavaScript* przywykorzystaniu środowiska *Node.js*.

1.2 Podział obowiązków

Marcin Nawrocki odpowiadał za opracowanie własnej biblioteki zawierającej operacje na obrazach w języku *Python*, a także stworzenie skryptu służącego do wywoływania odpowiednich operacji (API). **Tomasz Jankowski** natomiast był odpowiedzialny za opracowanie aplikacji reagującej na działania użytkownika i wywołującej odpowiednie instrukcje języka *Python*, a także stworzenie interfejsu graficznego oraz pliku *.exe* pozwalającego na uruchomienie aplikacji.

1.3 Zaimplementowane operacje

Konwersja do skali szarości zaimplementowana w dwoma sposobami. Z wykorzystaniem wzoru przystosowującą wagi kolorów do parametrów ludzkie oka. Wykorzystuje ona wzór Y = 0,215*R+0.7151*G+0.0721*B. Drugim sposobem jest obliczenie średnie z trzech składowych kolorowych (R+G+B)/3.

Progowanie (ang. *thresholding*) z manualnie ustalonym progiem w zakresie 1-54,

Automatyczne Progowanie (ang. *auto-thresholding*) z progiem obliczonym za pomocą metody Otsu.

Dylacja dwa elementy strukturalne do wyboru (kwadrat oraz krzyż) z regulowanym rozmiarem.

Erozja dwa elementy strukturalne do wyboru (kwadrat oraz krzyż) z regulowanym rozmiarem.

Otwarcie dwa elementy strukturalne do wyboru (kwadrat oraz krzyż) z regulowanym rozmiarem.

Domknięcie dwa elementy strukturalne do wyboru (kwadrat oraz krzyż) z regulowanym rozmiarem.

Filtr medianowy dwa elementy strukturalne do wyboru (kwadrat oraz krzyż) z regulowanym rozmiarem.

Filtr dolnoprzepustowy dostępne 4 jądra przekształcenia (LP1-LP4), przedstawione na rysunek 2.

Filtr górnoprzepustowy dostępne 4 jądra przekształcenia (HP1-HP4), przedstawione na rysunek 1.

Filtry górnoprzepustowe

HP1		
-1	-1	-1
-1	9	-1
-1	-1	-1

HP2		
0	-1	0
-1	5	-1
0	-1	0

HP3		
1	-2	1
-2	5	-2
1	-2	1

HP4		
0	-1	0
-1	20	-1
0	-1	0

Rysunek 1: Dostępne jądra przekształcenia dla operacji filtracji górnoprzepustowej.

LP1		
1	1	1
1	1	1
1	1	1

LP2		
1	1	1
1	2	1
1	1	1

LP3		
1	1	1
1	4	1
1	1	1

LP4		
1	1	1
1	12	1
1	1	1

Rysunek 2: Dostępne jądra przekształcenia dla operacji filtracji dolnoprzepustowej przepustowej.

Korekcja gamma z modyfikowalnym parametrem gamma.

Nakładanie szumu typu sól i pieprz z wyborem stosunku pieprzu do soli, oraz prawdopodobieństwem wystąpienia szumu na pojedyńczym pikselu (procent obrazu jaki zostanie pokryty szumem).

Nakładanie szumu gaussowskiego z możliwością wyboru wartości średniej oraz odchylenia standardowego.

Oprócz tego zaimplementowano funkcje pozwalające na odczyt kilku podstawowych parametrów obrazów:

histogram w przypadku zdjęć kolorowych oddzielny dla każdej składowej,

Minimalna i maksymalna wartość piksel dla każdej składowej,

Parametry statystyczne wartości pikseli - wariancja, odchylenie standardowe, mediana, średnia dla każdej składowej.

Oprócz tego niezbędne było napisanie kilku funkcji pomocniczych, odpowiadających przede wszystkim za generowanie między obrazów, wycinanie odpowiednich wycinków (realizacja różnych elementów strukturalnych), sprawdzanie palety barw zdjęcia itp.

2 Opis działania aplikacji

2.1 Przetwarzanie obrazów za pomocą języka Python

W celu efektywnego przetwarzania obrazów w języku Python wykorzystano trzy zewnętrzne biblioteki:

- NumPy,
- PIL,
- matplotlib.

Podstawową biblioteką dla tej aplikacji jest NumPy (ang.*Numerical Python*). Jest to biblioteka wspierająca działanie na wielowymiarowych tablicach i macierzach. Zdecydowano się na jej wykorzystanie, ponieważ implementacja jakichkolwiek obliczeń macierzowych na wbudowanych w Pythona kontenerach danych byłoby skrajnie nieoptymalne. Dlatego też wszystkie obrazy były przetwarzane jako tablice NumPy. Podczas wykorzystania tego typu danych kluczową kwestią jest wykonywanie jak największej ilości operacji za pomocą wbudowanych funkcji zamiast intuicyjne dokonywać iteracji po pikselu. Niestety nie dla każdej operacji było to możliwe, dlatego też niektóre z nich działają wyraźnie wolniej niż pozostałe.

Kolejną kluczową dla działania biblioteka była PIL (ang. *Python Image Library*). Wykorzystywany jest tylko jej podstawy moduł "Image". W naszym projekcie za jego pomocą odczytujemy oraz zapisujemy pliki graficzne. Zdecydowaliśmy się na tę bibliotekę ponieważ, umożliwia ona prostą konwersję z i do tablic NumPy.

Ostatnią wykorzystaną bibliotekę jest Matplotlib. Jest to potężne narzędzie służące do wizualizowania danych w języku Python. My jednak korzystaliśmy z małego wycinka jej możliwości, który pozwala na wyświetlanie w formie obrazów tablic NumPy, dzięki czemu w prosty sposób mogliśmy kontrolować poprawność działania naszych operacji. Biblioteka była wykorzystywana jedynie podczas tworzenia aplikacji, nie jest niezbędna do jej poprawnego działania.

2.2 Aplikacja okienkowa

2.2.1 Technologie

Node.js jest środowiskiem uruchomieniowym opartym o silnik *JavaScript V8* stworzony przez *Google*, który cechuje się wysoką wydajnością dzięki kompilacji plików źródłowych do kodu maszynowego przed ich wykonaniem. Środowisko to rozszerza możliwości języka JavaScript, dostarcza biblioteki umożliwiające szybsze i bardziej przystępne tworzenie aplikacji, ale przede wszystkim pozwala na uruchomienie kodu źródłowego poza przeglądarką. Służy głównie do tworzenia dynamicznych aplikacji internetowych (których treść zmienia się zależnie od działań użytkownika).

Electron.js to platforma programistyczna, która pozwala tworzyć aplikacje okienkowe (z interfejsem graficznym). W tym przypadku ta technologia została wykorzystana jako przeglądarka internetowa, w której wyświetlane są strony internetowe naipsane przy użyciu Node.js.

2.2.2 Struktura projektu

Ścieżka do opracowanego przez nas kodu źródłowego jest następująca:

```
resources/app/express-app/
```

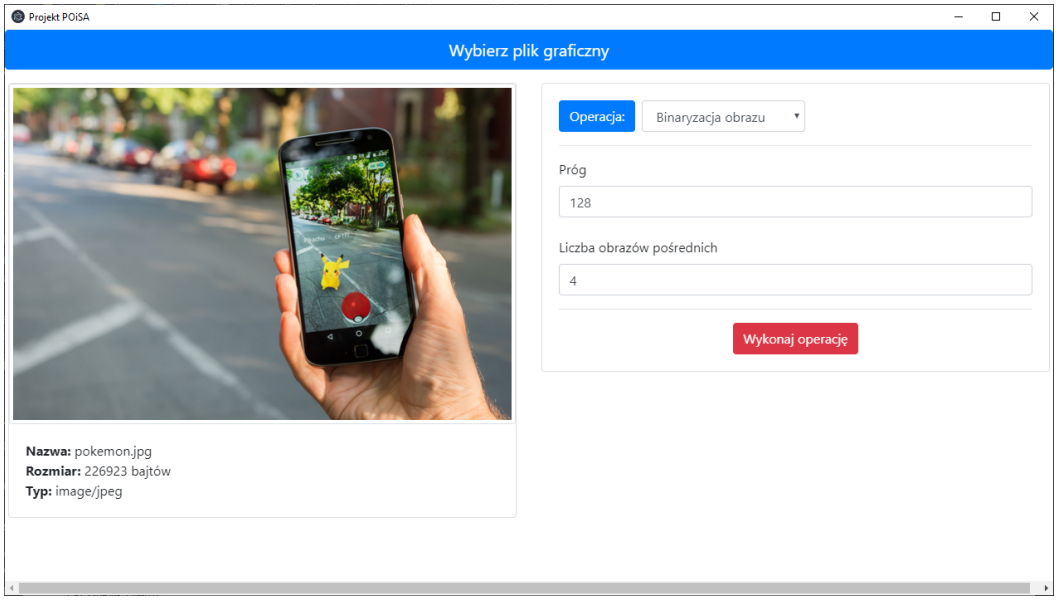
Projekt został podzielony na wiele folderów i plików, których znaczenie opisano poniżej. Pozostałe katalogi (niewymienione poniżej) zawierają domyślnie generowane pliki dla aplikacji okienkowej tego typu.

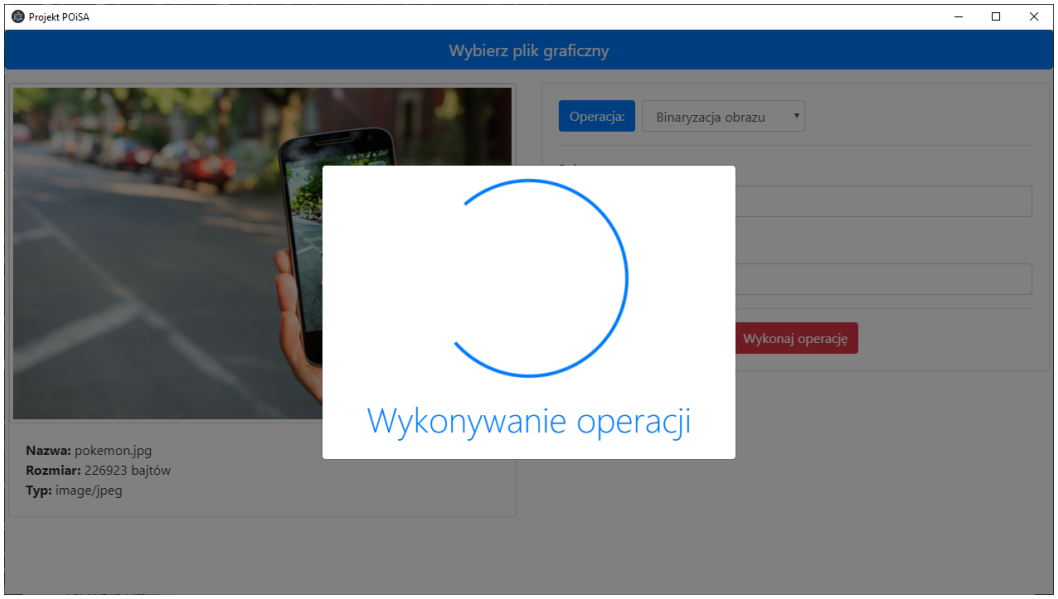
- *node_modules* – folder przechowujący zainstalowane pakiety.
- *public* – folder zawierający zasoby dostępne publicznie, dla dowolnego użytkownika – znajduje się tam kaskadowy arkusz stylów CSS (*style.css*) odpowiadający za wygląd strony internetowej, biblioteka *Bootstrap*, biblioteki *JavaScript* oraz skrypty napisane w języku *Python*.
- *views* – zawiera pliki opisujące interfejsy graficzne, tzn. strukturę konkretnych stron internetowych. W podfolderze *include* umieszczono nagłówek oraz stopkę, które są dołączane do każdej ze stron – dzięki zapisaniu powtarzalnych linii kodu w osobnych plikach unika się zjawiska redundancji.
- *app.js* – główny plik aplikacji, który definiuje sposób jej działania.
- *routes.js* – zawiera procedury wykonywane w przypadku odebrania konkretnych żądań, tj. wywołuje odpowiednie polecenia w wyniku poszczególnych działań użytkownika w aplikacji.
- *package.json* – plik przechowujący informacje o projekcie oraz pakietach, które powinny zostać zainstalowane w celu poprawnego działania aplikacji.

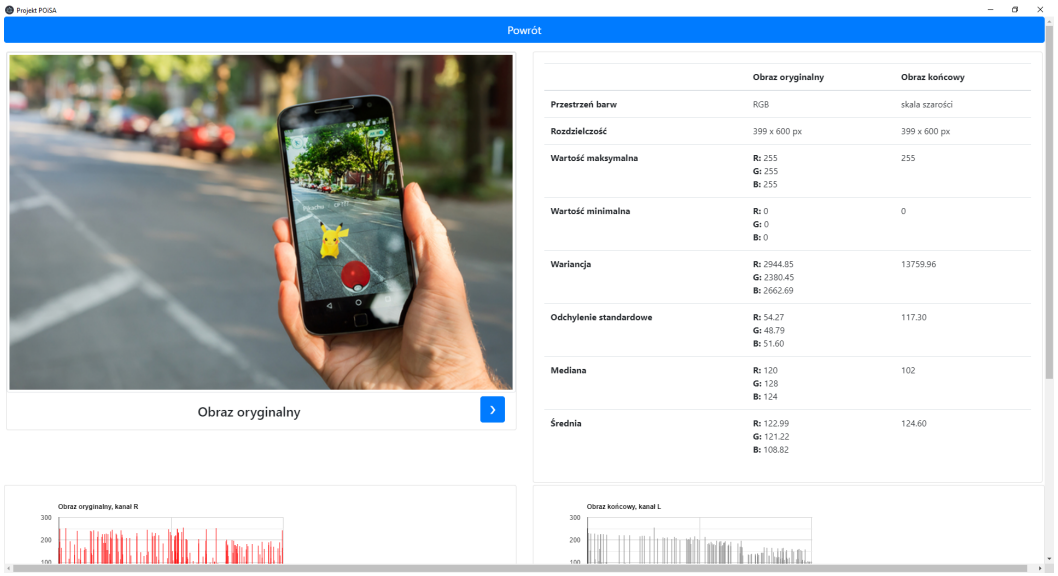
2.2.3 Interfejs graficzny

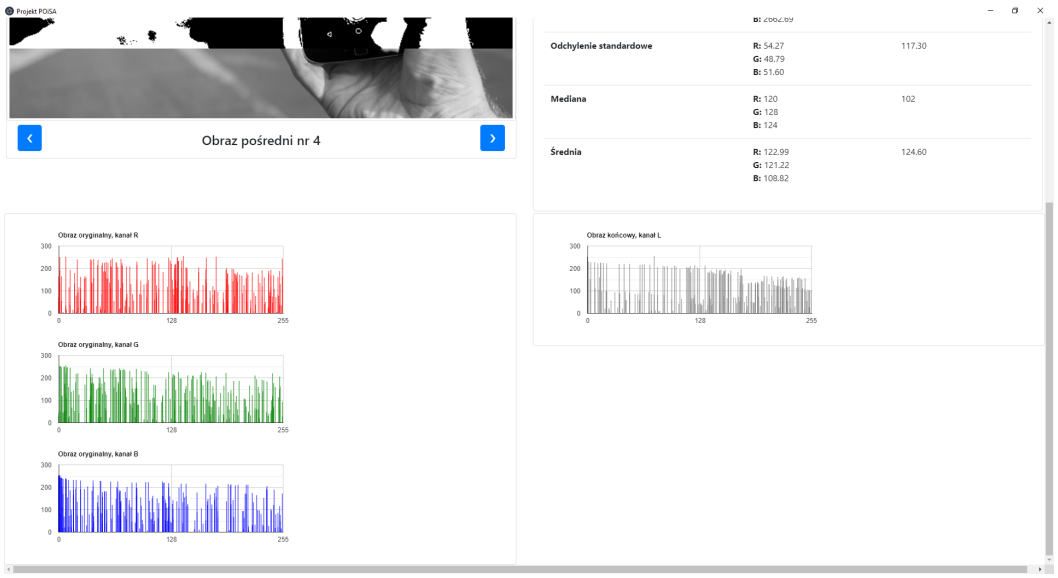
Wygląd wizualny aplikacji internetowej został opisany w językach HTML, CSS, JavaScript oraz EJS (*Embedded JavaScript templates*). Pierwsze trzy języki stanowią standard przy tworzeniu stron internetowych, natomiast czwarty z nich jest językiem, który służy do umieszczania danych dynamicznych, tj. pobranych z bazy danych. Dzięki niemu – przy użyciu składni języka JavaScript, na którym bazuje – można operować na danych jako tablicy, której każdy element jest wyświetlamy w sposób opisany językiem HTML.

2.3 Zrzuty ekranu z aplikacji




Projekt POiSA
Wybierz plik graficzny
Operacja:
Binaryzacja obrazu
Wykonaj operację
Wykonywanie operacji
Nazwa: pokemon.jpg
Rozmiar: 226923 bajtów
Typ: image/jpeg


Projekt POiSA
Powrót
Obraz oryginalny
Obraz końcowy
Przestrzeń barw
RGB
skala szarości
Rozdzielczość
399 x 600 px
399 x 600 px
Wartość maksymalna
R: 255
G: 255
B: 255
255
Wartość minimalna
R: 0
G: 0
B: 0
0
Wariancja
R: 2944.85
G: 2380.45
B: 2662.69
13759.96
Odchylenie standardowe
R: 54.27
G: 48.79
B: 51.60
117.30
Mediana
R: 120
G: 128
B: 124
102
Średnia
R: 122.99
G: 121.22
B: 108.82
124.60
Obraz oryginalny, kanał R
Obraz końcowy, kanał L


Projekt POiSA
Obraz pośredni nr 4
Odchylenie standardowe
R: 54.27
G: 48.79
B: 51.60
117.30
Mediana
R: 120
G: 128
B: 124
102
Średnia
R: 122.99
G: 121.22
B: 108.82
124.60
Obraz oryginalny, kanał R
Obraz końcowy, kanał L
Obraz oryginalny, kanał G
Obraz oryginalny, kanał B

3 Potencjalny rozwój aplikacji

3.1 Możliwe usprawnienia

Kierunki potencjalnego dalszego rozwoju aplikacji przedstawiono poniżej.

1. Generowanie większej liczby lub bardziej szczegółowych parametrów obrazów.
2. Rozwinięcie aplikacji o dodatkowe operacje na obrazach oraz dodanie nowych kształtów elementów strukturalnych (np.koło).
3. Możliwość jeszcze bardziej szczegółowego wpływania na wykonywane operacje (większa dostępna liczba parametrów).
4. Rozwój aplikacji w kierunku ogólnodostępnej strony internetowej.
5. Opracowanie plików wykonywalnych aplikacji również na platformy macOS oraz Linux.

3.2 Skalowalność

Dzięki skalowalnej strukturze biblioteki można w łatwy sposób rozwinąć jej możliwości. Przede wszystkim funkcja odpowiedzialna za filtrację dolno- i górno-przepustową może obsłużyć dowolne jądro przekształcenia, w dowolnym rozmiarze. Dlatego z łatwością możliwe jest dodanie kolejnych jąder przekształcenia np. wyostrzających krawędzie itp. Kolejnym przykładowym kierunkiem rozwoju byłoby dodanie nowego elementu strukturalnego, chociażby o okrągłym kształcie. Wymagało bo to rozszerzenie działania jednej funkcji a możliwe byłoby zastosowanie do każdej funkcji wykorzystującej elementy strukturalne.

Z punktu widzenia aplikacji rozwój aplikacji o nowe funkcjonalności również jest bardzo prosty. Dodanie kolejnych funkcjonalności ogranicza się do umieszczenia zaledwie kilku dodatkowych linii kodu w plikach odnoszących się do nowych funkcji. Odbywa się to w sposób analogiczny do już istniejących linii kodu dla obecnie zaimplementowanych operacji. Dla przykładu, przyjrzyjmy się sytuacji, w której chcielibyśmy dodać funkcję „newFunction", która przyjmuje dwa argumenty wejściowe.

```
case 'addSaltPepperNoise':
    options.args = ["addSaltPepperNoise", `${imagePath}`, `${body.
    addSaltPepperNoise.propability}`, `${body.addSaltPepperNoise.saltPepperRatio}
    `, `${body.addSaltPepperNoise.numberOfInters}`];
    break;

case 'newFunction':
    options.args = ["newFunction", `${imagePath}`, `${body.newFunction.arg1}`, `$
    {body.newFunction.arg2}`];
    break;
```

Rysunek 3: routes.js

```
'addSaltPepperNoise': addSaltPepperNoise,
'removeFiles': removeFiles,
'newFunction': newFunction
```

Rysunek 4: api.py

Po dodaniu funkcji do skryptu w języku *Python*, należy zadeklarować jej użycie w sposób przedstawiony na powyższych zrzutach ekranu (w odpowiednich miejscach kodu). Poza tym należy zaktualizować plik z widokiem strony internetowej „home.ejs" w sposób analogiczny do tam występującego (język *HTML*). Po tak prostych i niezajmujących wiele czasu modyfikacjach, nowa funkcja już będzie mogła zostać wywoływana z poziomu aplikacji okienkowej.