Programowanie w języku Java

Projekt

Narzędzie zwiększające głębię ostrości zdjęcia makro na podstawie serii fotografii makro z przesuwającą się płaszczyzną ostrości

https://github.com/PGszafir/Focus_Stacking_Java_01

W tym projekcie stworzyliśmy program stanowiący narzędzie, które na podstawie przykładowych serii zdjęć makro o małej głębi ostrości tworzy wynikowe zdjęcie makro z dużą głębią ostrości.

Główny rdzeń programu stanowi klasa ***Main*** która łączy w przejrzystą komplementarną całość pozostałe wyspecjalizowane klasy.

Po uruchomieniu programu pojawia się menu, w którym pytamy użytkownika, którą serię zdjęć mamy poddać przetwarzaniu. Użytkownik wpisuje wybrany, odpowiadający zdjęciom numer, po czym program przekazuje do swojej dalszej części niezbędne informacje, takie jak ścieżka do folderu ze zdjęciami, wygląd nazwy plików i ich liczbę.

Następnie program tworzy nazwę ścieżki do pierwszego zdjęcia i przekazuje ją do klasy ***FotoDecomposition.*** Wewnątrz tej klasy, za pomocą podstawowej metody wejścia/wyjścia, pobieramy plik z obrazem i umieszczamy w obiekcie klasy *BufferedImage*, która reprezentuje wczytany obraz w formie tablicy atrybutów odpowiadającym poszczególnym pikselom obrazka. Za jego pomocą dokonujemy wszelkich operacji na zdjęciach i ich pikselach. Gdy już utworzyliśmy odpowiedni obiekt tej klasy odpowiadający pierwszemu zdjęciu, przekazujemy do konkretnych zmiennych jego wymiary, takie jak szerokość i wysokość. Do pomocniczej tablicy za pomocą metody *getRGB* przekazujemy jednowymiarową tablicę pikseli w postaci integer w formie liczby systemu szesnastkowego 0xAARRGGBB, gdzie każde 2 kolejne cyfry szesnastkowe reprezentują wartość kolejnego kanału sRGB (A – kanał alfa, R – kanał czerwony, G – zielony, B – niebieski) w wartościach od 0 do 255 dziesiętnie. Ze względu na ograniczony obszar pamięci stosu nie przekształcaliśmy tych liczb do postaci tablic 4 elementowych short (najmniejszy możliwy typ do przedstawienia 255) tylko zdecydowaliśmy się na każdorazowe rzutowanie tej liczby do wartości konkretnego kanału na potrzeby konkretnych obliczeń w dalszej części projektu. Na końcu klasa przetransponowuje tablicę 1-wymiarową pikseli w docelową 2-wymiarową reprezentującą naturalnie rozumiane ich rozmieszczenie w wymiarze szerokości i wysokości. Daje to możliwość dojścia do każdego z nich poprzez 2 współrzędne reprezentujące wartość każdego z wymiarów, jak na osi x i y. Klasa *FotoDecomposition* poprzez swoją metodę zwraca tę dwuwymiarową tablicę pikseli w klasie głównej *Main*.

Otrzymany w ten sposób obraz przekazujemy do klasy ***GaussianBlur*** wykonującą rozmycie gaussowskie na pierwszym obrazie. Obiekt tej klasy sukcesywnie przechodzi przez wszystkie piksele obrazu, z lewej do prawy, nakładając standardową maskę mnożników $\begin{matrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{matrix}$ na badany piksel i wszystkie pozostałe go otaczające, uzyskując w wyniku jego nową „rozmytą” wartość. Robi to dla kanału zielonego ze względu na największą wydajność takiego zabiegu. Podobnie jak klasa *FotoDecomposition* zwraca ona tablicę 2-wymiarową wartości pikseli, oczywiście już zmienionych. Zwracana tablica 1. zdjęcia stanie się jednocześnie szablonem na którym będziemy dalej działać.

W następnej kolejności, pozostając przy zdjęciu nr 1, na podstawie jego mapy pikseli obliczamy dla każdego miejsca wskaźnik ostrości używając do tego metody Laplacian Edge Detection, czyli wykrywania krawędzi. W projekcie używamy do tego klasy ***EdgeDetection.*** Dla każdego piksela na obrazie oblicza ona poziom ostrości używając przy tym podobnej maski mnożników jak przy GaussianBlur, tylko o innych współczynnikach: $\begin{matrix}-1 & -1 & -1\\-1 & 8 & -1\\-1 & -1 & -1\end{matrix}$ i tym razem sumując wartości dla wszystkich 3 kolorów. Klasa ta za pomocą odpowiedniej metody zwraca 2-wymiarową tablicę wskaźników ostrości dla piksela znajdującego się w danym miejscu w obrazie. Ta tablica, tak samo jak poprzednia, będzie bazowym szablonem do wszelkich zmian przy wczytywaniu kolejnych zdjęć.

Po wykonaniu powyższych czynności program wchodzi do pętli, której liczba przejść jest wyznaczona ilością zdjęć w serii. Drugie zdjęcie przechodzi te same etapy przetwarzania co pierwsze zdjęcie (wczytanie w *FotoDecomposition*, rozmycie w *GaussianBlur* i obliczenie ostrości w *EdgeDetection*), uzyskane z niego tablice są jednak wczytywane do innych nowych zmiennych (oznaczanych w nazwie przedrostkiem „new”), w przeciwieństwie do poprzednich bazowych (nazwanych z przedrostkiem „current”).

Będąc cały czas w pętli i wczytawszy oba pierwsze zdjęcia przechodzimy do właściwego już zestawiania i nakładania zdjęć na siebie. Mapy pikseli wraz tablicą ostrości pierwszego zdjęcia (nazywane już później po prostu **bazowym**) razem z drugim (później będącym po prostu **nowym**) są wczytywane do obiektu klasy ***FocusStacking***. W tej klasie przechodząc kolejno przez wszystkie współrzędne obrazu, porównujemy wartości ostrości dla tych współrzędnych na obrazie i zostawiamy piksel na obrazie bazowym, jeśli wartość nowej ostrości (new) jest mniejsza od dotychczasowej (current) lub przenosimy wartość piksela do obrazu bazowego w przeciwnym przypadku. W celu częściowego wyeliminowania przesunięć zastosowaliśmy dodatkowy warunek, który musi spełnić piksel aby został wymieniony - jego różnica w wartościach kolorów od piksela dotychczasowego w tej lokalizacji nie może być większa niż empirycznie wyznaczona wartość progu (*COLOR_TRESHOLD*). Po przejściu przez wszystkie piksele, obiekt tej klasy za pomocą konkretnych metod zwraca do *Main* dwie nowe tablice – tablicę mapy pikseli i mapy ostrości

w tych pikselach. Zastępują one dotychczasowe tablice bazowe. To było przejście pierwsze pętli.

W kolejnych przejściach czynności są powtarzane dla kolejnych pobieranych zdjęć. Tablice każdego zdjęcia są zapisywane znowu jako te nowe (new) i są konfrontowane w klasie FocusStacking z tablicami bazowymi dając nowe, ulepszone tablice bazowe.

Po przejściu przez wszystkie zdjęcia pętla kończy się, a my dostajemy wynikową mapę pikseli, którą przekazujemy do klasy ***FotoRecomposition.*** Klasa ta tworzy obiekt klasy BufferedImage biorąc ścieżkę pierwszego zdjęcia, przekształca 2-wymiarową tablicę, którą otrzymała, do tablicy 1-wymiarowej kompatybilnej z metodą setRGB wspomnianej klasy i nakłada wynikowe, ostateczne wartości pikseli na buforowane zdjęcie (wymiary cały czas pozostają takie same, więc brak konfliktu). Na koniec klasa *FotoRecomposition* zapisuje buforowane zdjęcie wynikowe w formacie JPG w odpowiednim folderze projektu. Na tym program kończy swoje działanie.
Podczas tworzenia projektu natknęliśmy się na szereg problemów związanych m.in. z przesunięciami między zdjęciami i ich różnymi przybliżeniami, których niestety nie udało nam się całościowo skorygować metodami arytmetyczno-logicznymi.

Łukasz Jasielski
Jacek Bartoś
Piotr Gorczowski