Programowanie w Języku Python Podsumowanie Projektu

Szymon Rynkowski
Dominik Jastrząb-Moore

Link do repozytorium

link

Zastosowane biblioteki

1. Numpy - przyspieszenie i wykonywanie obliczeń oraz operacji na tablicach
2. Matplotlib - tworzenie wykresów
3. OpenCV - obsługa obrazów
4. PyQt - GUI

Temat projektu

Przedstawiony Opis Problemu:

1. Segmentacja obrazu na zadane obszary tonalne (np. światła, tony pośrednie i cienie) i stosowanie do nich osobnych wartości progowania automatycznego z manualną korektą.

Problem ma praktyczne zastosowanie przy np. przygotowywaniu reprodukcji rycin czy akwafort z materiałów o stosunkowo niewielkich rozdzielczościach. „Stosunkowo niewielka” rozdzielczość to rozdzielczość mniejsza niż ok. 200-300 PPI (ilustracja 1 - 01). Problem z taką rozdzielczością przy skanowaniu rycin polega na tym, że ryciny często zawierają na tyle subtelne detale (linie i punkty), które są przez skaner interpretowane nie binarnie, jako kolor papieru lub kolor farby drukarskiej, ale jako odcienie szarości. Ma to miejsce szczególnie w ciemnych fragmentach obrazu (w tzw. „cieniach”/„shadows”). Jeśli przy reprodukowaniu ustawić próg jasności, który wszystkie piksele jaśniejsze niż próg zamieni w białe, a ciemniejsze w czarne, w taki sposób, żeby detale w cieniach pozostały dobrze widoczne, to detale w „światłach” („highlights”) zaczną zanikać (ilustracja 1 - 03). I vice-versa: próg ustawiony dla świateł sprawi, że cienie zaczną stawać się jednolitą, czarną plamą (ilustracja 1 - 04). Nie pomoże resamplowanie obrazu, bo interpolacja po prostu rozciągnie wartości pikseli na piksele sąsiednie.

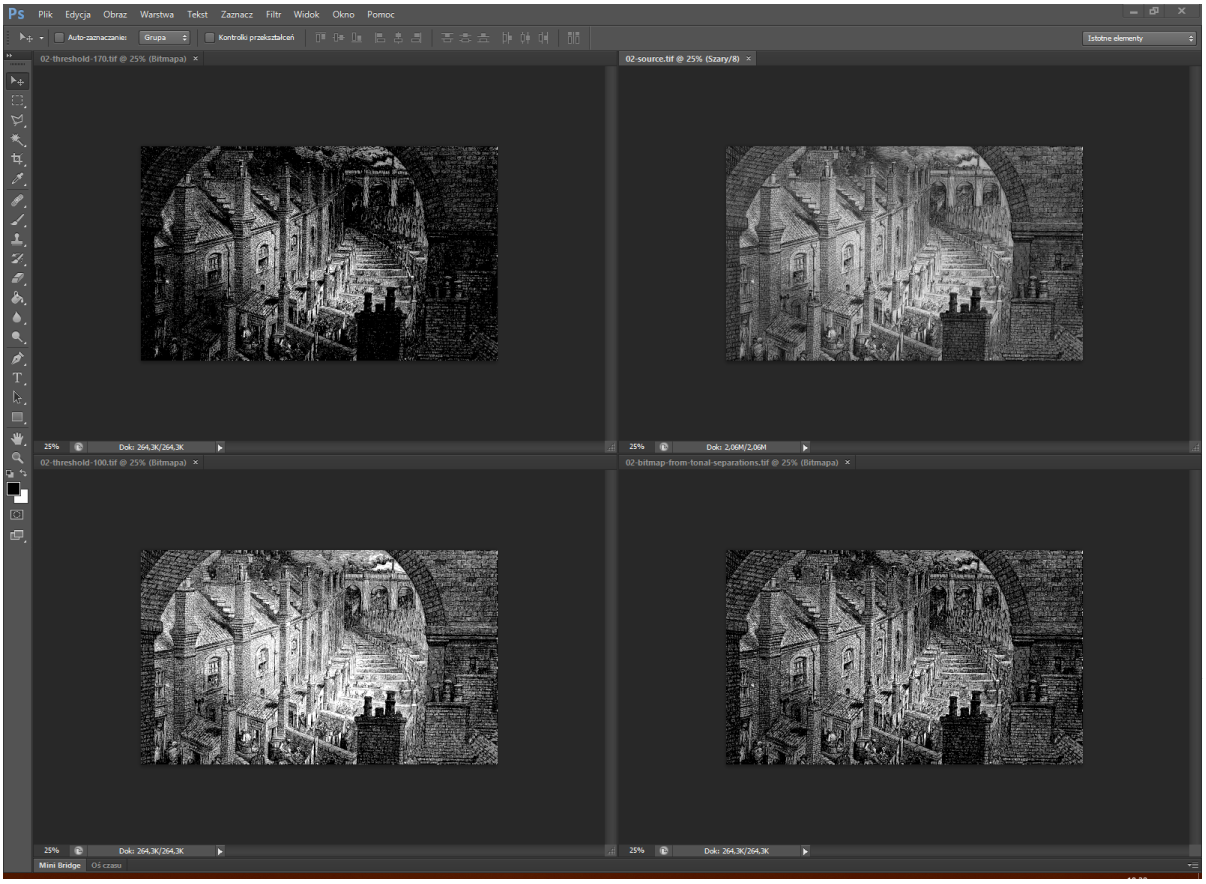
Stosunkowo prostą metodą poradzenia sobie z tym problemem jest segmentacja obrazu na obszary tonalne (światła, tony pośrednie i cienie to najpopularniejszy podział, ale można wyróżniać ile-kto-chce zakresów) i zaaplikowanie do każdego osobnego progu (i ewentualnie innych filtrów, np. morfologicznych). Chyba najprościej zacząć od potraktowania

obrazu filtrem dolnoprzepustowym, np. rozmyciem gaussowskim (ilustracja 1 - 05). To sprawi, że w obrazie łatwo będzie zaznaczyć „plamy” świateł, tonów pośrednich i cieni. Do każdego obszaru stosujemy różne wartości progowania i… ilustracja 1 - 06. Ilustracje 1 - 07 do 1 - 09 pokazują porównania obrazów źródłowego, sprogowanego dla świateł, dla cieni i z osobnymi progami dla każdego segmentu tonalnego.

Dla lepszej ilustracji wrzucam też przykładowe profile ciemnego (1 - 10) i jasnego (1 - 11) fragmentu. Jest też wersja z obydwoma profilami nałożonymi na siebie (1 - 12). Im wyższy schodek tym jaśniejszy obraz. Detale to miejsca pomiędzy „szczytami” a „dolinami”. Widać z tych profilów, że jeśli „poziomą kreskę” (progowanie) ustawić tak, żeby przecinała miejsca pomiędzy szczytami a dolinami dla świateł (czerwony profil), to „przeleci” ona nad szczytami dla cieni (niebieski profil) sprawiając, że tamte staną się czarne.

I jeszcze gwoli wytłumaczenia dlaczego w ogóle używać progowania do reprodukcji, skoro obraz w skali szarości wygląda fajnie. Owszem, on wygląda fajnie na monitorze, gdzie i tak (sprawdzić czy nie „retina”) nie ma szału z rozdzielczością. Ba! Nawet w druku będzie dość przyzwoicie, ALE… w druku można uzyskać znacznie więcej. Jeśli wydrukujesz obraz w skali szarości, to RIP (Raster Image Processor) w drukarki rozłoży każdy piksel na większe i mniejsze punkty, żeby zasymulować półtony skali szarości. Natomiast jeśli dasz mu gotową bitmapę, to zwyczajnie naniesie ją piksel po pikselu na płytę. Różnica? W pierwszym przypadku zagubią się nieco detale, a krawędzie będą sprawiały wrażenie odrobinę nieostrych. W drugim krawędzie będą ostre i kontrastowe. Różnica jest do wychwycenia nawet dla laika, bo choć może nie wiedzieć dokładnie jak co nazwać, to dostrzeże, że „jest lepiej” (sprawdzone przy składaniu materiałów pokonferencyjnych z archeologii).

2. Progowanie przy nierównym oświetleniu strony.



Które problemy rozwiązuje projekt

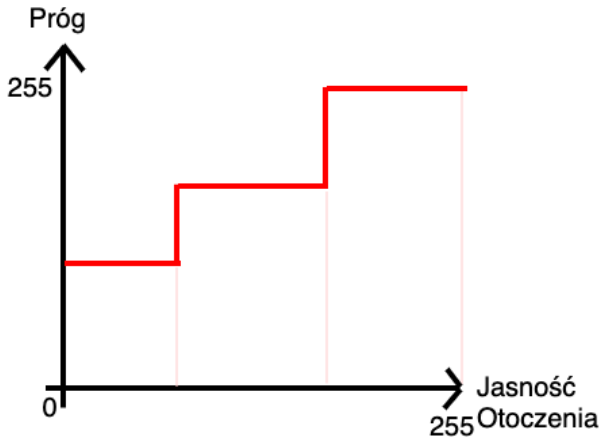
Nasz projekt skupi się na adaptacyjnym progowaniu obrazów monochromatycznych - zarówno skanów dokumentów jak i rycin.

Rozwiązane problemy

Jak rozwiązać problem przedstawiony w opisie za pomocą progowania adaptacyjnego.

zaproponowane przez autora problemu rozwiązanie wymaga pobierania od użytkownika zarówno przedziałów jak i progów. sprawia to, że uzyskanie dobrego rozwiązania wymaga dużego nakładu pracy.

Poniżej przedstawiam wykres funkcji progu od jasności otoczenia piksela dla 3 przedziałów:



Aby zwiększyć dokładność metody należy zwiększyć ilość przedziałów. Wiąże się to jednak ze zwiększeniem ilości danych które trzeba pobrać od użytkownika.

Zauważmy, że gdy będziemy zwiększać ilość przedziałów uzyskamy funkcje coraz bardziej przypominającą zwyczajną ciągłą funkcję (oraz coraz większą dokładność).

Nowym problemem jest sposób pobierania takiej funkcji od użytkownika.

Dobór funkcji

Głównym problemem w progowaniu adaptacyjnym jest dobór funkcji progu od jasności otoczenia dla określonego przypadku.

Funkcja będzie pobierana od użytkownika, jednak ważne jest aby podawało się ją w prosty, szybki i łatwy w modyfikacji sposób

Sposób pobierania funkcji

1. interpolacja
w bardziej skomplikowanych przypadkach będziemy od użytkownika pobierać wartość funkcji progu w paru z góry ustalonych punktach, a funkcje uzyskujemy za pomocą interpolacji

2. stała b funkcji liniowej postaci y = x + b
w prostych przypadkach (głównie progowanie tekstu z lekkimi zaburzeniami) nie potrzeba zbyt skomplikowanej funkcji progu - wystarczy taka jak wyżej.
Od użytkownika pobieramy jedynie wartość stałej b.

Sposób określania jasności otoczenia

Zostanie wykorzystane rozmycie Gaussa - jasność piksela na rozmytym obrazie jest równoznaczna z jasnością otoczenia.
Od użytkownika trzeba pobrać zasięg otoczenia.

Interakcja z użytkownikiem

Zastosujemy GUI

Pobieranie wartości od użytkownika

Ponieważ wszystkie wartości które musimy pobrać mieszczą się w łatwym do przewidzenia z góry zakresie zastosujemy slidery.
Umożliwia to łatwą, szybką i intuicyjną zmianę parametrów.

Ocena poprawności podanych parametrów

Aby stwierdzić czy przy danych parametrach progowanie działa zgodnie z oczekiwaniami udostępnimy użytkownikowi podgląd wyniku na żywo.

Stwarza to dodatkowy problem - operacja progowania musi wykonywać się bardzo szybko inaczej GUI nie będzie płynne.

Optymalizacja operacji progowania

Można zauważyć, że potrzebna nam jest wartość funkcji progu jedynie dla 256 argumentów. możemy te wartości stablicować i nie liczyć funkcji osobno dla każdego piksela.

Po zastosowaniu operacja progowania wykonuje się bardzo szybko oraz poziom skomplikowania funkcji propagującej nie wpływa mocno na wydajność.
Oznaczo to, że można zwiększyć ilość pobieranych parametrów jeżeli użytkownik potrzebuje większej dokładności.
Zwiększenie ilości punktów oraz zastosowanie metody interpolacji która minimalizuje wpływ zmiany wartości w punkcie na punkty odległe od zmienionego pozwoli uzyskać prawie dowolną funkcję - więc pozwoli sprogować dobrze dowolny obrazek.

Dodatkowe udogodnienia dla użytkownika

1. tryb wsadowy
2. obsługa wielu formatów obrazu - zarówno przy odczycie jak i zapisie.
3. podgląd aktualnego wykresu funkcji progu
4. różne tryby interpolacji

Podsumowanie

W rezultacie uzyskaliśmy system zdolny do progowania zarówno skanów rycin jak i dokumentów tekstowych. Dzięki zastosowanej optymalizacji system jest łatwo rozszerzyć o nowe, dokładniejsze metody progujące w zależności od potrzeb użytkownika.