Podstawy Grafiki Komputerowej
Selektywna desaturacja obrazu
Projekt nr 046

Tomasz Madej
Julia Bała
Dawid Wojdylak

12 czerwca 2021

1 Opis Projektu

Celem projektu było napisanie programu, który umożliwia wczytanie plików typu *.jpg*, a następnie edytowanie go poprzez przeprowadzenie częściowej desaturacji obrazu według wybranych kryteriów:

- jasność - w kolorze pozostaną jedynie jasne elementy,
- składowa R(Red) koloru w modelu RGB,
- składowa G(Green) koloru w modelu RGB,
- składowa B(Blue) koloru w modelu RGB,
- wartość koloru(Hue) - na podstawie odcienia,
- składowa C(Cyan) koloru w modelu CMY,
- składowa M(Magenta) koloru w modelu CMY,
- składowa Y(Yellow) koloru w modelu CMY.

Aplikacja umożliwia ustawienie stopnia desaturacji oraz poziomu rozmycia parametru. Zakres zmienianych wartości wynosi dla Hue od 0^o do 360^o, natomiast dla pozostałych kryteriów od 0 do 255. Program umożliwia również zapis obrazu do pliku typu *.jpg*.

2 Podział pracy i analiza czasowa

Cały projekt wykonano wspólnie. Na początku stworzono okno główne umożliwiające wczytanie, zapis oraz wyświetlenie obrazu. Następnie zaimplementowano okno opcji. Ostatnim i najbardziej zaawansowanym zadaniem było stworzenie klasy odpowiedzialnej za desaturację oraz jej warianty.

3 Analiza kodu

Do stworzenia programu wykorzystano język C++ oraz framework Qt. Prace wykonywano w środowisku Qt Creator.

3.1 *mainwindow.h*

Podstawą programu jest klasa *MainWindow* dziedzicząca po klasie *QMainWindow* (z biblioteki Qt). Zawiera ona składowe potrzebne do utworzenia bazowego okna, które widzimy tuż po uruchomieniu programu.

Składowe klasy *MainWindow*:

- void resizeEvent(QResizeEvent* event); - metoda reagująca na zmianę rozmiaru okna, dostosowuje rozmiary i proporcje obrazów
- void setImage(); - metoda wyświetlająca obrazy
- void setImageChanged(); - metoda wyświetlająca zmieniony obraz
- void on_actionOpen_triggered(); - metoda otwierająca okno wyboru obrazu
- void on_actionSave_triggered(); - metoda zapisująca zmieniony obraz
- void on_actionOptions_triggered(); - metoda otwierająca okno ustawień
- void on_actionAbout_triggered(); - metoda wyświetlająca okno informacji o twórcach aplikacji
- DesatProcess * _desaturation; - wskaźnik do obiektu klasy *DesatProcess* (opisanej poniżej) służący do procesu desaturacji
- QPixmap _original; - oryginalny obraz
- float _relation; - relacja jednego boku obrazu do drugiego
- float _relationPanel; - relacja jednego boku panelu do drugiego

3.2 *options.h*

Klasa *Options* zawarta w pliku *options.h* odpowiedzialna jest za obsługę okna ustawień desaturacji.

Składowe klasy *Options*:

- void on_treshold_valueChanged(int value); - metoda reagująca na zmianę wartości progu desaturacji

- void on_saturation_valueChanged(int value); - metoda reagująca na zmianę poziomu saturacji
- void on_blur_valueChanged(int value); - metoda reagująca na zmianę wartości rozmycia
- void on_close_clicked(); - metoda wyłączająca okno opcji
- void on_typeSat_activated(int index); - metoda reagująca na zmianę kryterium desaturacji

3.3 *desatprocess.h*

Klasa *DesatProcess* jest istotą projektu. W niej zawarte są funkcje przeprowadzające proces selektywnej desaturacji programu.

Składowe klasy *DesatProcess*:

- void load(QPixmap buffer); - metoda wczytująca obraz, jako argument przyjmuje obiekt typu *QPixmap*, który reprezentuje obraz w bibliotece Qt.
- void refresh(); - metoda odświeżająca i przeprowadzająca proces desaturacji
- QPixmap getPixmap() const; - metoda zwracająca obraz po procesie desaturacji
- void setTresh(int val); - metoda ustawiająca próg desaturacji
- void setBlur(int val); - metoda ustawiająca wartość rozmycia
- void setSat(int val); - metoda ustawiająca wartość saturacji w %
- void setTypeSat(int val); - metoda ustawiająca kryterium desaturacji
- int getTresh() const - getter zwracający parametr progu desaturacji;
- int getBlur() const - getter zwracający parametr rozmycia
- int getSat() const - getter zwracający parametr saturacji
- int getTypeSat() const - getter zwracający kryterium desaturacji
- int _teshold; - próg desaturacji
- int _blur; - rozmycie
- int _saturaion; - poziom saturacji
- int _typeSat; - kryterium desaturacji
- QImage _original; - oryginalny obraz
- QImage _changed; - obraz po desaturacji
- int minCol(const int & red, const int & green, const int & blue) const; - metoda zwracająca najmniejszą składową wartość koloru RGB

- int maxCol(const int & red, const int & green, const int & blue, bool cmy = false, const int & dark = 0) const; - metoda zwracająca największą składową wartość koloru RGB
- int midCol(const QColor & col) const; - metoda zwracająca środkową wartość składową koloru RGB
- void lightness(const QColor & col, int & min, int & max, int & temp) const; - metoda przygotowująca parametr temp do desaturacji na podstawie jasności
- void hue(const QColor & col, int & min, int & max, int & temp) const; - metoda przygotowująca parametr temp do desaturacji na podstawie odcienia
- int checkAngle(const QColor & col,const int & max, const int & min) const; - metoda wyliczająca kąt koloru na kole barw na potrzebę wyliczenia *hue*.

4 Opis Algorytmu

Główny algorytm przeprowadzający desaturację obrazu znajduje się w metodzie void refresh(); klasy *DesatProcess*. Na początku sprawdzane jest, w jakiej reprezentacji koloru będzie przeprowadzana desaturacja: RGB czy CMY. Dalsza część jest podobna, lecz działa na innych składowych kolorów. Przechodząc pętlą po każdym pikselu sprawdzane jest, jaka opcja została wybrana i w zależności od tego przygotowana jest zmienna *temp*, która jest wartością całkowitą składowych koloru lub wartości hue:

1. jasność - uruchamiana jest metoda, w której wartość temp jest wyliczana jako średnia arytmetyczna wartości maksymalnej i minimalnej składowych koloru.
2. wartość koloru (hue) - na początku sprawdzana jest wartość kąta położenia koloru na kole barw wykorzystując operacje warunkowe i wartość maksymalną i minimalną koloru. Jeżeli wartość maksymalna składowej koloru jest równa minimalnej to wartość temp ustawiana jest na 0, w innym przypadku jako ((mid - min) / (max - min) * 60) + angle, gdzie mid to wartość pośrednia składowych koloru.
3. składowa R koloru - temp = wartość składowej R.
4. składowa G koloru - temp = wartość składowej G.
5. składowa B koloru - temp = wartość składowej B.
6. składowa C koloru - temp = wartość składowej C.
7. składowa M koloru - temp = wartość składowej M.
8. składowa Y koloru - temp = wartość składowej Y.

Następnie jeżeli wartość temp jest mniejsza od różnicy ustawionej wartości progu desaturacji do ustawionej wartości rozmycia, korzystamy z algorytmu częściowej desaturacji który wylicza nam wartości składowe koloru, które ustawiamy. W algorytmie tym obliczamy różnice składowych *deltaR, deltaG, deltaB*, jako wyliczona wartość maksymalna składowych koloru pomniejszona o wartość składową R,G,B, a następnie przemnożona razy wartość (1 - ustawiony procent saturacji/100).

Ustawiony kolor ma wartość ($R+deltaR, G+deltaG, B+deltaB$).

W innym przypadku jeśli wartość temp jest większa od różnicy ustawionej wartości progu desaturacji i ustawionej wartości rozmycia i wartość temp jest mniejsza od ustawionego progu desaturacji przeprowadzamy "rozmytą desaturację", która jest przeprowadzana jak powyżej z wykorzystaniem częściowej desaturacji. Dodatkowo sprawdzamy, czy poszczególna wyliczona delta jest większa od 0. Jeśli tak to modyfikujemy ją w następujący sposób: deltaR * (1-delta(wyliczona jako próg desaturacji - wartość temp + poziom desaturacji) / ustawiony poziom rozmycia) i tworzymy kolor piksela tak jak powyżej.
W innej sytuacji nie zmieniamy piksela.

5 Instrukcja użycia programu

Po uruchomieniu programu należy wybrać zdjęcie klikając przycisk *Open*, który znajduje się w pasku Menu w lewym górnym rogu. Po wybraniu i załadowaniu obrazu powinien on się ukazać w oknie głównym w dwóch wariantach. Po lewej stronie widzimy obraz oryginalny, natomiast po prawej obraz poddany procesowi desaturacji.

Aby zmienić opcje desaturacji należy kliknąć przycisk *Options* z paska Menu. Pojawi się małe okno, które zawiera ustawienia desaturacji. Na samej górze wybieramy opcję z listy *Saturaion point*. Następnie ustawiamy stopień saturacji w %. Poniżej znajduje się opcja *Set Value*, która reguluje próg saturacji. Na samym dole za pomocą suwaka *Set Blur* dostosowujemy rozmycie, które wygładza przejście pomiędzy pikselami poddanymi desaturacji oraz o oryginalnej barwie.

6 Testowanie

Przy testowaniu sprawdzaliśmy zgodność wyniku z założeniami teoretycznymi.
Oto kilka naszych wyników:

- Dla ustawień jasności:

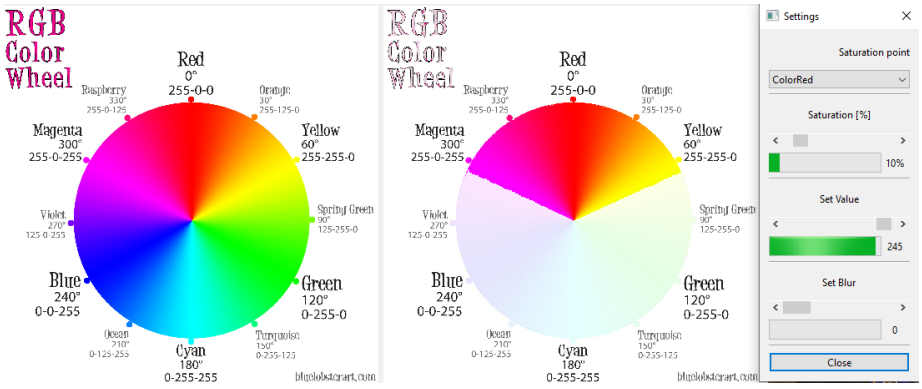
- Dla ustawień wartość koloru(Hue)

- Dla koloru niebieskiego:

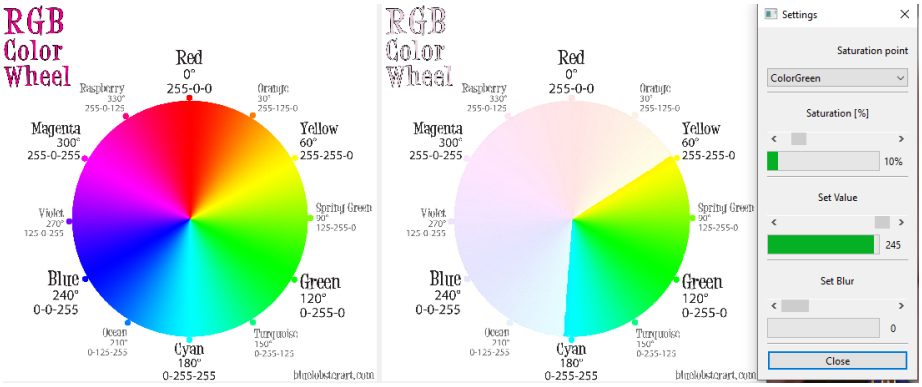
Można dostrzec, że pierwszym kolorem, który "zanika", jest kolor żółty. Składowe żółtego to RGB(255,255,0). Zatem widać, że algorytm zadziałał zgodnie z oczekiwaniami.

Testy dla każdej opcji desaturacji przeprowadzono na kole kolorów:

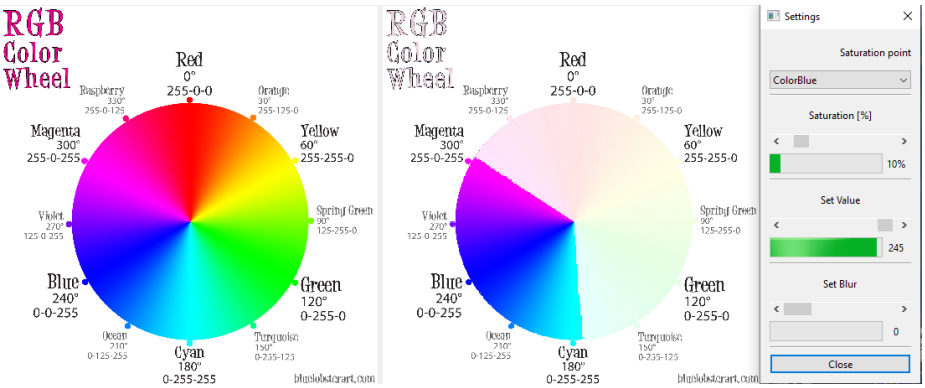
- ColorRed



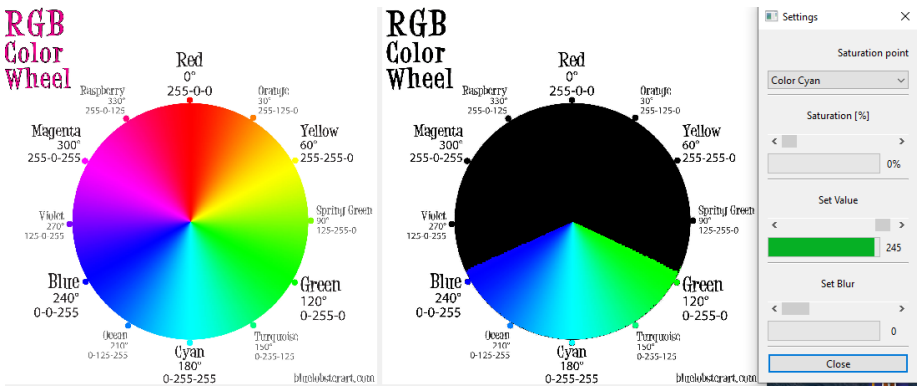
- ColorGreen



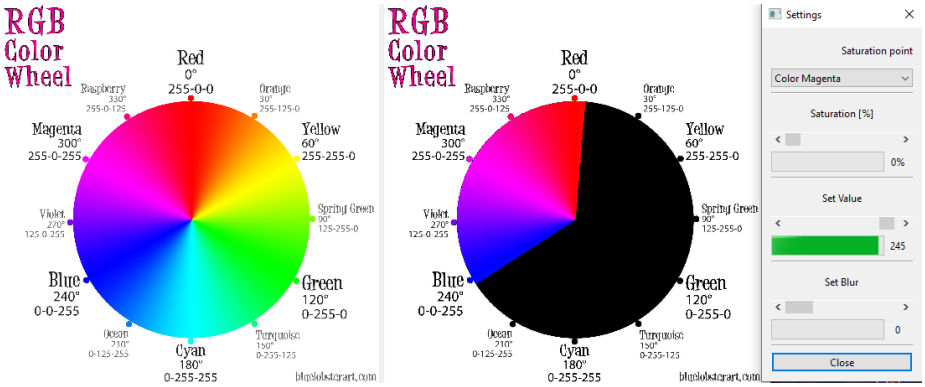
- ColorBlue



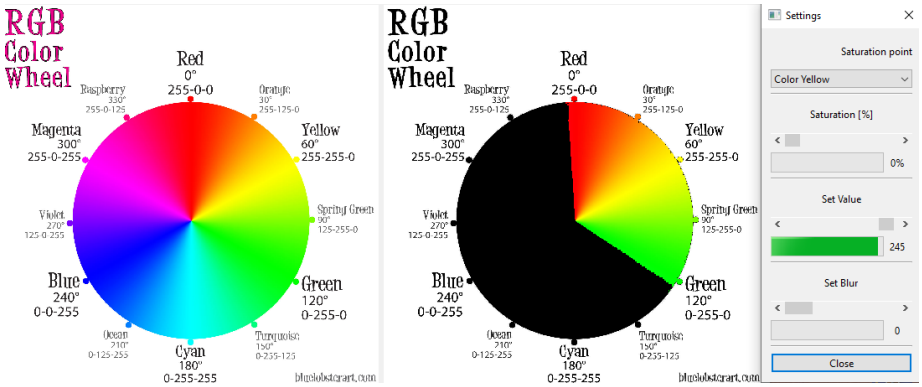
- Color Cyan



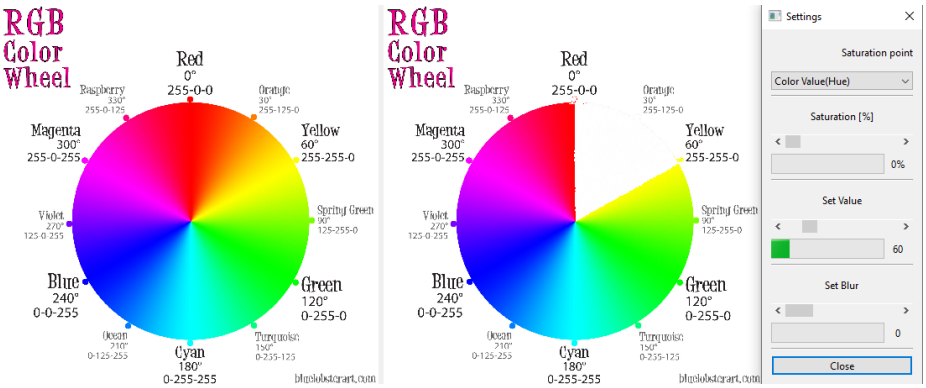
- Color Magenta



- Color Yellow



- Hue dla 60^o



7 Wdrożenie, raport i wnioski

Program uruchamiano i testowano z różnymi zdjęciami. Pierwszym problemem napotkanym w przypadku zdjęć o wysokiej rozdzielczości była niska wydajność programu. Dla zdjęć o niskiej rozdzielczości program był wydajny. Jeżeli chodzi o sam efekt częściowej desaturacji, to program spisuje się bardzo dobrze. Otrzymane zdjęcia po odpowiednim doborze parametrów wyglądają efektownie.

W dalszym rozwijaniu programu należałoby się zająć optymalizacją wydajności algorytmów, aby praca ze zdjęciami o wysokiej rozdzielczości była płynna i komfortowa. Najprostszym możliwym rozwiązaniem (choć niekoniecznie wygodnym) byłoby zaimplementowanie przycisku, który uruchamiałby algorytmy odpowiedzialne za desaturację tylko raz (w obecnej wersji programu metoda `void refresh()`; uruchamiana jest w trakcie każdej interakcji z kontrolkami). Niestety nie udało nam się zaimplementować rozmycia Gaussowskiego, co również byłoby pożądane przy dalszym rozwoju projektu.