HW5

*Date: 2021. 10. 14*

*Student ID: 21600635*

*Name: Bomoon JUNG*

1. **Homework 5**

1) Develop a program that edits the input image

* Read ‘lena.png’ an image as a color image. Depending on keyboard input of a user, your program should do the following operations.
* Negative transformation on the user input ‘n’
* Gamma transformation with the value of gamma as 2.5 on the user input ‘g’
* Histogram equalization on the user input ‘h’
* Reset the image on the user input ‘r’
* Read ‘colorful.jpg’ an image as a color image. Depending on keyboard input of a user, your program should do the following operations.
* Color slicing on the user input ‘s’

>> Hue value: 9 < hue < 23

* Color conversion on the user input ‘c’

>> Increase Hue value by 50

>> For hue values bigger than 129, subtract 129 instead

* Reset the image on the user input ‘r’
* Read ‘balancing.jpg’ an image as a color image. Depending on keyboard input of a user, your program should do the following operations.
* Average filtering on the user input ‘a’

>> Use ‘blur’ function with mask size as 9x9

* White balancing by using gray world assumption on the user input ‘w’
* Reset the image on the user input ‘r’

2) Your program should display three windows

* ‘lena’, ‘color’, ‘balancing’
* Depending on the input of a user, contents in a window should be changed.
* For color conversion, use CV\_BGR2HSV and CV\_HSV2BGR
* Use waitKey for user interaction
* waitKey returns the code of the pressed key or 1 if no key was pressed before the specified time had elapsed
* ESC -> 27

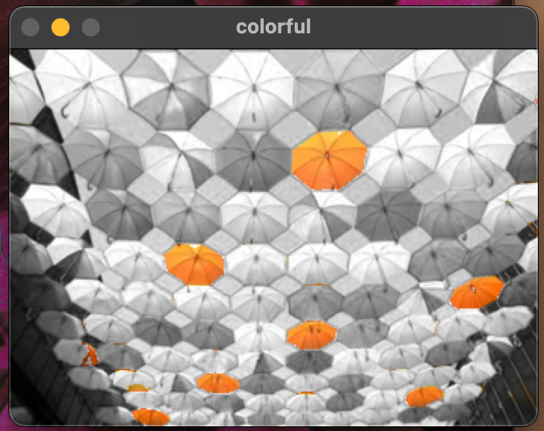
텍스트, 실내, 사람, 자주색이(가) 표시된 사진

자동 생성된 설명**텍스트, 사람이(가) 표시된 사진

자동 생성된 설명**텍스트, 사람이(가) 표시된 사진

자동 생성된 설명

**Figure 1. option of ‘n’, ‘g’ and ‘h’**

실외, 다채로운, 디스플레이이(가) 표시된 사진

자동 생성된 설명

Figure 2. option of ‘s’ and ‘c’

텍스트, 실내이(가) 표시된 사진

자동 생성된 설명**텍스트, 실내이(가) 표시된 사진

자동 생성된 설명**

Figure 3. option of ‘a’ and ‘w’

2) explanation

In my code, display the original image at first time. After that program get option for ‘lena.jpg’, ‘colorful.jpg’ and ‘balacing.jpg’ respectively from user. In stage of getting option, if user input some other non-option about image, program get option again from user. when all option is entered, program display the changed image and ask to what user going to do. If user input ‘ESC’, the program will quit. Otherwise, program reenter the option.

3) Source code

*// g++ $(pkg-config --cflags --libs opencv) HW5\_21600635.cpp -o ./a*

*#include* <opencv2/opencv.hpp>

*#include* <iostream>

*#include* <string>

using *namespace* cv ;

using *namespace* std ;

Mat

negativeTransformation ( Mat src ) {

Mat negative\_img = src.clone() ;

Mat HSV ;

vector<Mat> hsv(3) ;

cvtColor(negative\_img, HSV, CV\_BGR2HSV) ; *// 에러 나옴*

split(HSV, hsv) ;

*int* pixel\_size = hsv[2].rows \* hsv[2].cols \* hsv[2].channels() ;

*for* ( *int* i = 0 ; i < pixel\_size ; i++) {

hsv[2].data[i] = 255 - hsv[2].data[i] ;

}

merge(hsv, negative\_img) ;

cvtColor(negative\_img, negative\_img, CV\_HSV2BGR) ;

*return* negative\_img ;

}

Mat

gammaTransformation ( Mat src ) {

Mat gamma\_img = src.clone() ;

Mat HSV ;

vector<Mat> hsv(3) ;

cvtColor(gamma\_img, HSV, CV\_BGR2HSV) ;

split(HSV, hsv) ;

*int* pixel\_size = hsv[2].rows \* hsv[2].cols \* hsv[2].channels() ;

*unsigned* *char* look\_up[256] ;

*int* arr\_size = sizeof(look\_up) / sizeof(*unsigned* *char*) ;

*float* gamma = 2.5 ;

*for* ( *int* i = 0 ; i < arr\_size ; i++ ) {

look\_up[i] = saturate\_cast<uchar>( pow( i / 255.0, gamma) \* 255.0) ;

}

*for* (*int* i = 0 ; i < pixel\_size ; i++ ) {

hsv[2].data[i] = look\_up[ hsv[2].data[i] ] ;

}

merge(hsv, gamma\_img) ;

cvtColor(gamma\_img, gamma\_img, CV\_HSV2BGR) ;

*return* gamma\_img ;

}

Mat

equalizeHistogram( Mat src ) {

Mat equal\_img = src.clone() ;

Mat HSV ;

vector<Mat> hsv(3) ;

cvtColor(equal\_img, HSV, CV\_BGR2HSV) ;

split(HSV, hsv) ;

equalizeHist(hsv[2], hsv[2]) ;

merge(hsv, equal\_img) ;

cvtColor(equal\_img, equal\_img, CV\_HSV2BGR) ;

*return* equal\_img ;

}

Mat

lena\_transformation ( Mat src ) {

Mat image ;

*while* ( true ) {

*char* option ;

cout << "Please press the option for lena image." << endl ;

cout << "'n' is Negative transformation, 'g' is Gamma transformation, 'h' is Histogram equalization, 'r' is reset." << endl ;

option = waitKey(0) ;

*if* ( option == 'n' ) { *// n == 110*

image = negativeTransformation( src ) ;

*break* ;

}

*else* *if* ( option == 'g' ) { *// g == 103*

image = gammaTransformation( src ) ;

*break* ;

}

*else* *if* ( option == 'h' ) { *// h == 104*

image = equalizeHistogram( src ) ;

*break* ;

}

*else* *if* ( option == 'r' ) { *// r == 114*

image = src.clone() ;

*break* ;

}

*else* { *// wrong option*

cout << "Wrong option. please press option key.1" << endl ;

*continue* ;

}

}

*return* image ;

}

Mat

colorSlicing( Mat src ) {

Mat slicing\_img = src.clone() ;

Mat HSV ;

vector<Mat> hsv(3) ;

cvtColor(slicing\_img, HSV, CV\_BGR2HSV) ;

split(HSV, hsv) ;

*// equalizeHist(hsv[2], hsv[2]) ;*

*int* pixel\_size = hsv[2].rows \* hsv[2].cols \* hsv[2].channels() ;

*for*( *int* i = 0 ; i < pixel\_size ; i++ ) {

*if*( hsv[0].data[i] > 9 && hsv[0].data[i] < 23 )

hsv[1].data[i] = hsv[1].data[i] ;

*else*

hsv[1].data[i] = 0 ;

}

merge(hsv, slicing\_img) ;

cvtColor(slicing\_img, slicing\_img, CV\_HSV2BGR) ;

*return* slicing\_img ;

}

Mat

colorConversion( Mat src ) {

Mat conv\_img = src.clone() ;

Mat HSV ;

vector<Mat> hsv(3) ;

cvtColor(conv\_img, HSV, CV\_BGR2HSV) ;

split(HSV, hsv) ;

*int* pixel\_size = hsv[2].rows \* hsv[2].cols \* hsv[2].channels() ;

*for*( *int* i = 0 ; i < pixel\_size ; i++ ) {

*if* ( hsv[0].data[i] > 129)

hsv[0].data[i] = hsv[0].data[i] + 50 - 129 ;

*else*

hsv[0].data[i] = hsv[0].data[i] + 50 ;

}

merge(hsv, conv\_img) ;

cvtColor(conv\_img, conv\_img, CV\_HSV2BGR) ;

*return* conv\_img ;

}

Mat

colorful\_transformation( Mat src ){

Mat image ;

*while* ( true ) {

*char* option ;

cout << "Please press the option for colorful image." << endl ;

cout << "'s' is Color slicing, 'c' is Color conversion, 'r' is reset." << endl ;

option = waitKey(0) ;

*if* ( option == 's' ) {

image = colorSlicing( src ) ;

*break* ;

}

*else* *if* ( option == 'c' ) {

image = colorConversion( src ) ;

*break* ;

}

*else* *if* ( option == 'r' ) {

image = src.clone() ;

*break* ;

}

*else* { *// wrong option*

cout << "Wrong option. please press option key." << endl ;

*continue* ;

}

}

*return* image ;

}

Mat

averageFiltering( Mat src ) {

Mat ave\_img = src.clone() ;

blur(ave\_img, ave\_img, Size(9, 9)) ;

*return* ave\_img ;

}

Mat

whiteBalancing( Mat src ) {

Mat white\_img = src.clone() ;

Mat BGR[3] ;

split(white\_img, BGR) ;

*int* pixel\_size = white\_img.rows \* white\_img.cols ;

*for*( *int* c = 0 ; c < white\_img.channels() ; c++ ) {

*int* sum = 0 ;

*for*( *int* i = 0 ; i < pixel\_size ; i++ ) {

sum += BGR[c].data[i] ;

}

*double* avg = sum / pixel\_size ;

*for*( *int* i = 0 ; i < pixel\_size ; i++ ) {

*int* temp = (128 / avg) \* BGR[c].data[i] ;

*if*( temp > 255)

BGR[c].data[i] = 255 ;

*else*

BGR[c].data[i] = temp ;

}

}

merge(BGR, 3, white\_img) ;

*return* white\_img ;

}

Mat

balancing\_transformation( Mat src ) {

Mat image ;

*while* ( true ) {

*char* option ;

cout << "Please press the option for balancing image." << endl ;

cout << "'a' is Average filtering, 'w' is White balancing, 'r' is reset." << endl ;

option = waitKey(0) ;

*if* ( option == 'a' ) {

image = averageFiltering( src ) ;

*break* ;

}

*else* *if* ( option == 'w' ) {

image = whiteBalancing( src ) ;

*break* ;

}

*else* *if* ( option == 'r' ) {

image = src.clone() ;

*break* ;

}

*else* { *// wrong option*

cout << "Wrong option. please press option key." << endl ;

*continue* ;

}

}

*return* image ;

}

*int*

main () {

Mat lena = imread("./resources/lena.png") ;

Mat colorful = imread("./resources/colorful.jpg") ;

Mat balancing = imread("./resources/balancing.jpg") ;

*// exception*

*if* ( lena.empty() || colorful.empty() || balancing.empty()) {

cout << "no such file" << endl ;

*return* 0 ;

}

imshow("lena", lena) ;

imshow("colorful", colorful) ;

imshow("balancing", balancing) ;

*while* ( 1 ) {

*// lena*

Mat lena\_ = lena\_transformation( lena ) ;

*// colorful*

Mat colorful\_ = colorful\_transformation( colorful ) ;

*// balancing*

Mat balancing\_ = balancing\_transformation( balancing ) ;

*// show the image*

destroyAllWindows() ;

imshow("lena", lena\_) ;

imshow("colorful", colorful\_) ;

imshow("balancing", balancing\_) ;

*// terminate case*

cout << "If you want to quit, press 'ESC' to terminate. If you want to continue, press any other key" << endl ;

*if* ( waitKey(0) == 27 ) {

*break* ;

}

}

*return* 0 ;

}