

HW5 Report

Computer Vision
Section 02
21800181 Kim Jisu

● HW 5

- Purpose

This is a program that read images "lena.png", "colorful.jpg", "balancing.jpg" and change the display image depending on the input.

- Principle

```
void cv::equalizeHist(InputArray src, OutputArray dst )
```

Equalizes the histogram of a grayscale image.

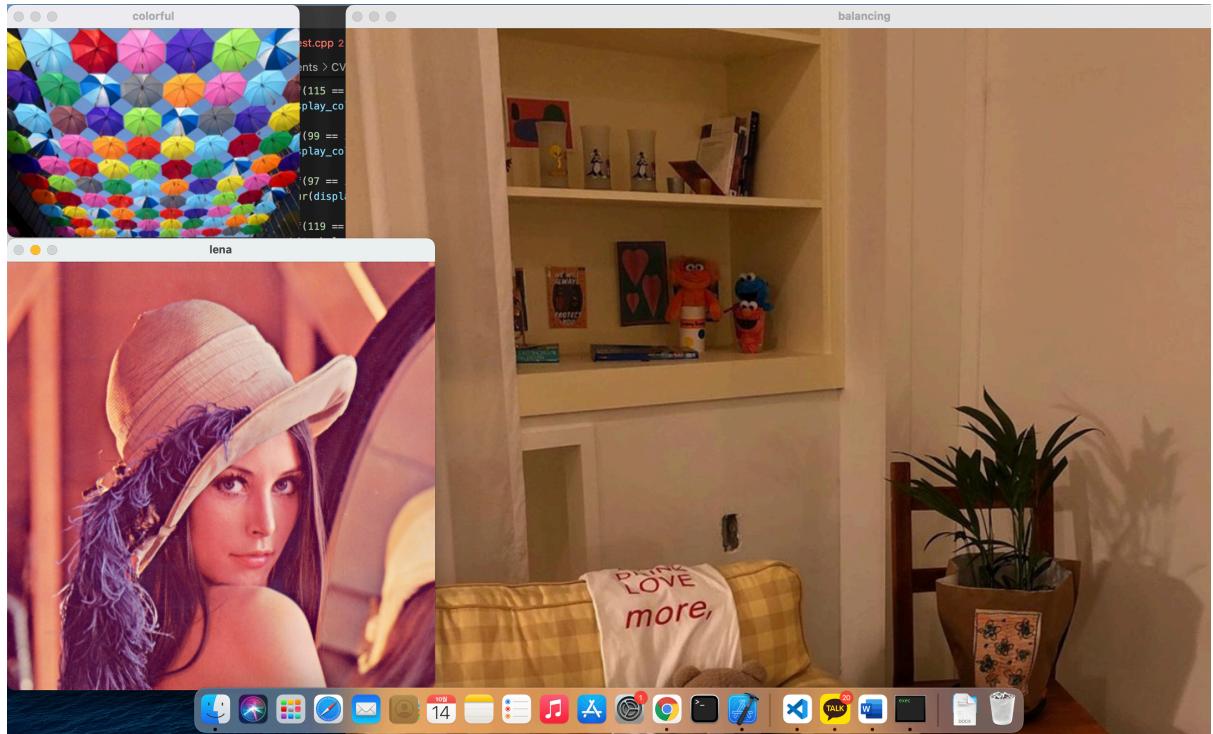
```
void cv::merge(const Mat * mv, size_t count, OutputArray dst )
```

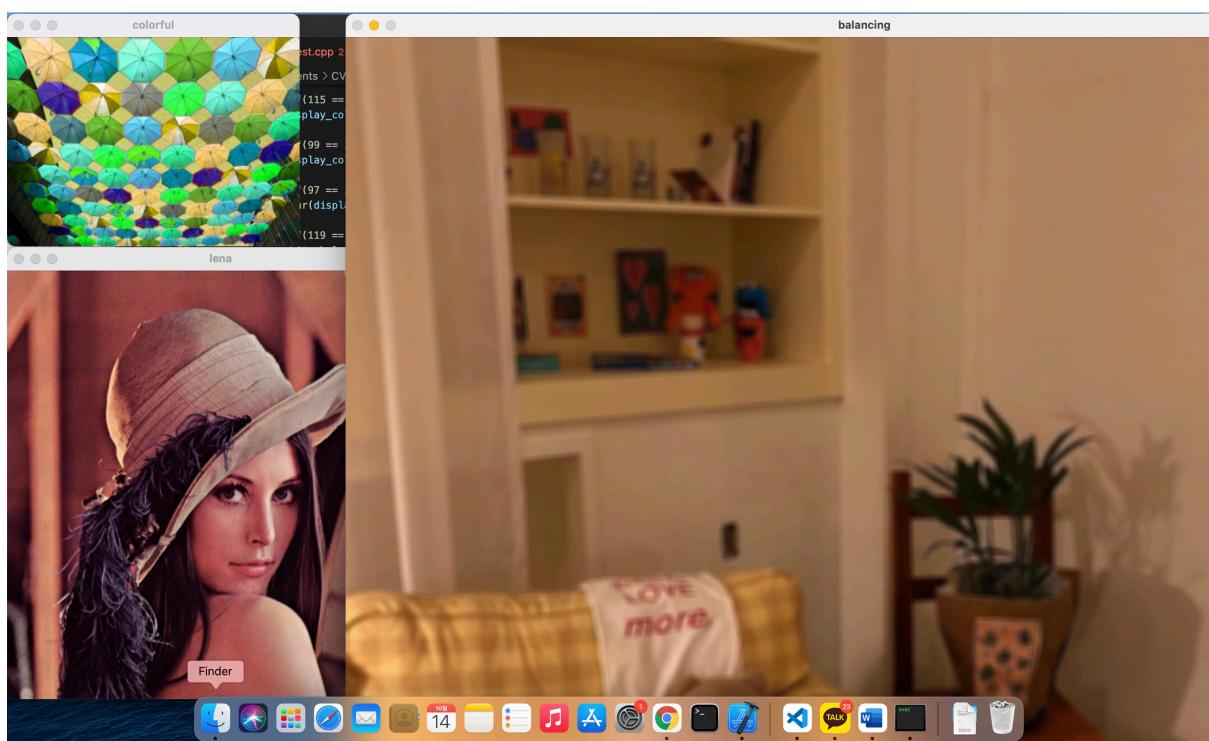
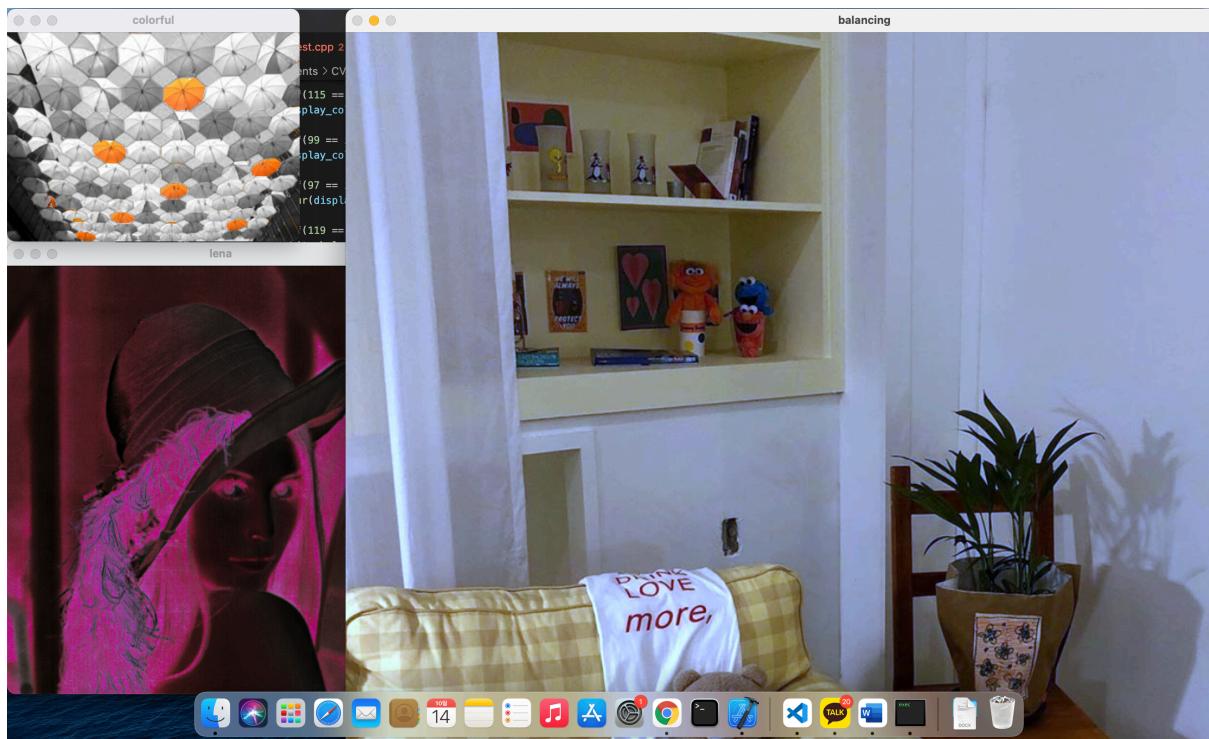
Creates one multi-channel array out of several single-channel ones.

```
void cv::split(const Mat & src, Mat * mvbegin )
```

Divides a multi-channel array into several single-channel arrays.

- Result





- Code

```
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace std;
using namespace cv;
Mat negative(Mat img);
Mat gamma(Mat img);
```

```
Mat hist(Mat img);

void white_balancing_g(Mat img);
Mat conversion(Mat img);
Mat color_slicing(Mat img);

int main () {
Mat lena,colorful,balancing; //r
Mat colorful_slicing, colorful_conversion; //s c
Mat display_lena, display_colorful,display_balancing;

lena = imread("lena.png");
colorful = imread("colorful.jpg");
balancing = imread("balancing.jpg");

display_lena = lena.clone();
display_colorful = colorful.clone();
display_balancing = balancing.clone();

int input;

while(true){
imshow("lena",display_lena);
imshow("colorful",display_colorful);
imshow("balancing",display_balancing);
input = waitKey(30000);

if(27 == input || -1 == input) break; //ESC
else if(114 == input){ //r
display_lena = lena.clone();
display_colorful = colorful.clone();
display_balancing = balancing.clone();
}
else if(110 == input){ //n
display_lena = negative(display_lena);
}
else if(103 == input){ //g
display_lena = gamma(display_lena);
}
else if(104 == input){ //h
display_lena = hist(display_lena);
}
else if(115 == input){ //s
display_colorful = color_slicing(display_colorful);
}
else if(99 == input){ //c
display_colorful = conversion(display_colorful);
}
else if(97 == input){ //a
blur(display_balancing,display_balancing,Size(9,9));
}

}
```

```

else if(119 == input){ //w
white_balancing_g(display_balancing); }
}

Mat negative(Mat img){
    cvtColor(img, img, CV_BGR2HSV);
    for(int i = 0;i <img.rows;i++)
        for(int j = 0; j<img.cols;j++)
            img.at<Vec3b>(i,j)[2] = 255 - img.at<Vec3b>(i,j)[2];
    cvtColor(img, img, CV_HSV2BGR);
return img; }

Mat gamma(Mat img){
float gamma = 2.5;
cvtColor(img, img, CV_BGR2HSV);

for(int i = 0;i <img.rows;i++)
for(int j = 0; j<img.cols;j++)
img.at<Vec3b>(i,j)[2] = pow((float)(img.at<Vec3b>(i,j)[2]/255.0),gamma)*255.0f;
convertScaleAbs(img,img,1,0);
cvtColor(img, img, CV_HSV2BGR);
return img;
}

Mat hist(Mat img){
vector<Mat> HSV_channel;
cvtColor(img, img, CV_BGR2HSV);
split(img,HSV_channel);
equalizeHist(HSV_channel[2], HSV_channel[2]);
merge(HSV_channel,img);
cvtColor(img, img, CV_HSV2BGR);
return img;
}

void white_balancing_g(Mat img) {
Mat bgr_channels[3];
int sum_b = 0 , sum_g = 0,sum_r = 0;
int avg_b, avg_g, avg_r;
split(img, bgr_channels);

for(int i = 0;i<img.rows;i++){
    for(int j = 0; j<img.cols; j++){
        sum_b += bgr_channels[0].at<uchar>(i,j);
        sum_g += bgr_channels[1].at<uchar>(i,j);
        sum_r += bgr_channels[2].at<uchar>(i,j);
    }
}

avg_b = sum_b / (img.rows * img.cols);
avg_g = sum_g / (img.rows * img.cols);

```

```

avg_r = sum_r / (img.rows * img.cols);

for(int i = 0; i<img.rows; i++)
for(int j = 0; j < img.cols; j++) {
int b = (128*bgr_channels[0].at<uchar>(i, j)/avg_b);
if(b > 255) bgr_channels[0].at<uchar>(i, j) = 255;
else bgr_channels[0].at<uchar>(i, j) = b;
int g = (128*bgr_channels[1].at<uchar>(i, j)/avg_g);
if(g > 255) bgr_channels[1].at<uchar>(i, j) = 255;
else bgr_channels[1].at<uchar>(i, j) = g;
int r = (128*bgr_channels[2].at<uchar>(i, j)/avg_r);
if(r > 255) bgr_channels[2].at<uchar>(i, j) = 255;
else bgr_channels[2].at<uchar>(i, j) = r;
}

merge(bgr_channels, 3, img);
}

Mat conversion(Mat img){
cvtColor(img, img, CV_BGR2HSV);
for(int i = 0; i<img.rows; i++){
    for (int j = 0; j < img.cols; j++) {
        if(img.at<Vec3b>(i,j)[0] > 129) img.at<Vec3b>(i,j)[0] -=129;
        else img.at<Vec3b>(i,j)[0]+=50;
    }
}
cvtColor(img, img, CV_HSV2BGR);
return img;
}

Mat color_slicing(Mat img){
cvtColor(img, img, CV_BGR2HSV);
for(int i = 0; i<img.rows; i++){
    for (int j = 0; j < img.cols; j++) {
        if(img.at<Vec3b>(i,j)[0]>9 && img.at<Vec3b>(i,j)[0]<23)
            img.at<Vec3b>(i,j)[1] = img.at<Vec3b>(i,j)[1];
        else img.at<Vec3b>(i,j)[1] = 0;
    }
}
cvtColor(img, img, CV_HSV2BGR);
return img;
}

```

- Video

https://youtu.be/iPY_Ueq60hE

- References

<https://docs.opencv.org/master/>