histogram_eq_technical_report

introduction

input 이미지를 histogram equalization하고, histogram equalization에 사용된 전이함수와 결과 이미지의 PDF, 오리지널 이미지의 PDF를 plot한다.

Grayscale image 관련 코드 구현

hist_eq.cpp

main 함수

```
1)
  Mat input = imread("input.jpg", IMREAD_COLOR);
    Mat input_gray;

    cvtColor(input, input_gray, COLOR_RGB2GRAY);    // convert RGB to Graysca

    Mat equalized = input_gray.clone();

    // PDF or transfer function txt files
    FILE *f_PDF;
    FILE *f_equalized_PDF_gray;
    FILE *f_trans_func_eq;

    fopen_s(&f_PDF, "PDF.txt", "w+");
    fopen_s(&f_equalized_PDF_gray, "equalized_PDF_gray.txt", "w+");
    fopen_s(&f_trans_func_eq, "trans_func_eq.txt", "w+");

2)
    float *PDF = cal_PDF(input_gray);    // PDF of Input image(Grayscale) : [L]
    float *CDF = cal_CDF(input_gray);    // CDF of Input image(Grayscale) : [L]

    G trans_func_eq[L] = { 0 };           // transfer function
```

```
3)
    hist_eq(input_gray, equalized, trans_func_eq, CDF);                    // histo
    float *equalized_PDF_gray = cal_PDF(equalized);                        // equa

    for (int i = 0; i < L; i++) {
        // write PDF
        fprintf(f_PDF, "%d\t%f\n", i, PDF[i]);
        fprintf(f_equalized_PDF_gray, "%d\t%f\n", i, equalized_PDF_gray[i]);

        // write transfer functions
        fprintf(f_trans_func_eq, "%d\t%d\n", i, trans_func_eq[i]);
    }
```

1. input 이미지 로드, rgb를 grayscale로 변환

   histogram equalization을 할 이미지를 오리지널 이미지를 clone해서 생성
2. grayscale로 변환된 오리지널 이미지의 PDF 계산

   grayscale로 변환된 오리지널 이미지의 CDF 계산
3. histogram equalization 진행

```
void hist_eq(Mat &input, Mat &equalized, G *trans_func, float *CDF) {

1)
    // compute transfer function
    for (int i = 0; i < L; i++)
        trans_func[i] = (G)((L - 1) * CDF[i]);

2)
    // perform the transfer function
    for (int i = 0; i < input.rows; i++)
        for (int j = 0; j < input.cols; j++)
            equalized.at<G>(i, j) = trans_func[input.at<G>(i, j)];
}
```

   1. transfer function 계산

$$T(r) = (L-1)CDF_r(r)$$

해당 공식을 사용한다.

grayscale의 input이미지의 CDF을 이용해 이미지 intensity 조절.

2. transfer function 적용

   equalized될 이미지의 픽셀에, 인풋 이미지에 trans_func을 적용해서 변환된 값들을 넣어준다.

equalized_trans.py

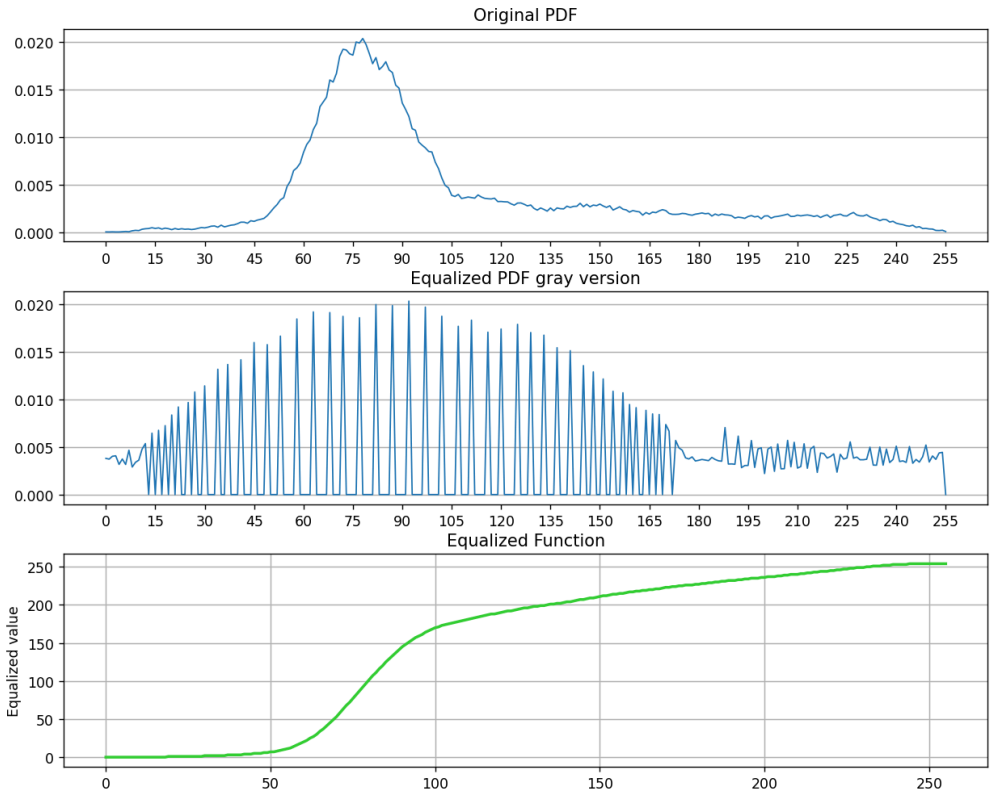
```
import matplotlib.pyplot as plt
import numpy as np

# 데이터 불러오기
pdf = np.loadtxt("PDF.txt", delimiter="\t", usecols=1)
eq_pdf = np.loadtxt("equalized_PDF_gray.txt", delimiter="\t", usecols=1)
trans_func = np.loadtxt("trans_func_eq.txt")
x = np.arange(256)
y_trans = trans_func[:, 1]


fig, axs = plt.subplots(3, 1, figsize=(10, 10))

# Original PDF
axs[0].plot(x, pdf, linewidth=1)
axs[0].set_title("Original PDF")
axs[0].set_xticks(range(0, 256, 15))
axs[0].grid(True, axis='y')

# Equalized PDF
axs[1].plot(x, eq_pdf, linewidth=1)
axs[1].set_title("Equalized PDF gray version")
axs[1].set_xticks(range(0, 256, 15))
axs[1].grid(True, axis='y')

# Equalized Function
axs[2].plot(x, y_trans, linewidth=2, color='limegreen')
axs[2].set_title("Equalized Function")
```

```
axs[2].set_xlabel("Original value")
axs[2].set_ylabel("Equalized value")
axs[2].grid(True)


plt.tight_layout()
plt.show()
```

오리지널 이미지의 PDF, Stretching을 거친 이미지의 PDF, 전이함수를 시각적으로 표현하는 python code

Result

plot한 결과는 다음과 같다.



원본 이미지의 PDF

PDF 분석 결과, 대부분의 intensity가 90과 105 사이에 몰려있다. 그렇기 때문에 픽셀이 중간보다 약간 어두운 영역에 몰려있고, 전체적인 이미지의 대비가 낮다고 볼 수 있다.

histogram equalization 된 이미지의 PDF

밝기가 70 ~ 100에 몰려 있었는데 전체적으로 평활화 되었다.

x축이 0~ 255 전 영역을 꽉 채우고 있어, 이미지가 더 넓은 명암을 가지게 되었다.

Stretching function

linear한 함수 형태로 나타난다

결론

차례로 input 이미지와 histogram equalization 진행한 이미지다.

대비가 향상되고, 전체적으로 더 선명해졌지만, 디테일이 살짝 부족해졌다는 것을 알 수 있다.

==========

RGB image 관련 코드 구현

hist_eq_RGB.cpp

```
1)
   Mat input = imread("input.jpg", IMREAD_COLOR);
   Mat equalized_RGB = input.clone();

2)
   float **PDF_RGB = cal_PDF_RGB(input); // PDF of Input image(RGB) : [L][3]
   float **CDF_RGB = cal_CDF_RGB(input); // CDF of Input image(RGB) : [L][3]

   G trans_func_eq_RGB[L][3] = { 0 };      // transfer function

3)
   // histogram equalization on RGB image
   hist_eq_Color(input, equalized_RGB, trans_func_eq_RGB, CDF_RGB);
   // equalized PDF (RGB)
   float** equalized_PDF_RGB = cal_PDF_RGB(equalized_RGB);
```

```
for (int i = 0; i < L; i++) {
    for (int c = 0; c < 3; c++) {
        fprintf(f_equalized_PDF_RGB, "%d\t%f\n", i + c * L, equalized_PDF_RG
        fprintf(f_PDF_RGB, "%d\t%f\n", i + c * L, PDF_RGB[i][c]);
        fprintf(f_trans_func_eq_RGB, "%d\t%d\n", i + c * L, trans_func_eq_RGB
    }
}
```

1. input 이미지 불러오기, output 이미지로 저장할 equalized_RGB 객체는 input clone해서 쓰기
2. input 이미지의 PDF, CDF 계산하기

   <hist_func.h>

```
float **cal_PDF_RGB(Mat &input) {

    int count[L][3] = { 0 };
    float **PDF = (float**)malloc(sizeof(float*) * L);

    for (int i = 0; i < L; i++)
        PDF[i] = (float*)calloc(3, sizeof(float));

    //Count
    for (int i = 0; i < input.rows; i++) {
        for (int j = 0; j < input.cols; j++) {
            for (int c = 0; c < 3; c++) {
                count[input.at<Vec3b>(i, j)[c]][c]++;
            }
        }
    }

    //Compute PDF
    for (int i = 0; i < L; i++) {
        for (int c = 0; c < 3; c++) {
            PDF[i][c] = (float)count[i][c] / (float)(input.rows * input.cols);
        }
    }
```

```
    return PDF;
}
```

일반 PDF 계산과정과 흡사하다.

하지만, RGB 이미지의 경우, 각각의 세 채널을 고려해야하기 때문에, 이중 for문을 사용하여 모든 채널과 모든 픽셀에 접근해 계산한다.

```
float **cal_CDF_RGB(Mat &input) {

    int count[L][3] = { 0 };
    float **CDF = (float**)malloc(sizeof(float*) * L);

    for (int i = 0; i < L; i++)
        CDF[i] = (float*)calloc(3, sizeof(float));

    // Count
    for (int i = 0; i < input.rows; i++) {
        for (int j = 0; j < input.cols; j++) {
            for (int c = 0; c < 3; c++) {
                count[input.at<Vec3b>(i, j)[c]][c]++;
            }
        }
    }

    // Compute CDF
    for (int i = 0; i < L; i++) {
        for (int c = 0; c < 3; c++) {
            CDF[i][c] = (float)count[i][c] / (float)(input.rows * input.cols);

            if (i != 0)
                CDF[i][c] += CDF[i - 1][c];
        }
    }

    return CDF;
}
```

일반 CDF 계산 과정과 흡사하다.

하지만, RGB 이미지의 경우, 각각의 세 채널을 고려해야하기 때문에, 이중 for문을 사용하여 모든 채널과 모든 픽셀에 접근해 계산한다.

3. histogram equalization을 진행한다.

```
void hist_eq_Color(Mat &input, Mat &equalized, G(*trans_func)[3], float **C
    // compute transfer function
    for (int c = 0; c < 3; c++) {
        for (int i = 0; i < L; i++) {
            trans_func[i][c] = (G)((L - 1) * CDF[i][c]);
        }
    }

    // perform the transfer function
    for (int i = 0; i < input.rows; i++) {
        for (int j = 0; j < input.cols; j++) {
            for (int c = 0; c < 3; c++) {
                equalized.at<Vec3b>(i, j)[c] = trans_func[input.at<Vec3b>(i, j)[c]
            }
        }
    }
}
```

일반적인 histogram equalization과 코드는 비슷하나, RGB 이미지 이기 때문에 for (int c = 0; c < 3; c++)를 사용하여 세 R, G, B 채널에 각각 접근해야한다.

equalized_RGB_trans.py

```
import matplotlib.pyplot as plt
import numpy as np

# 데이터 불러오기
pdf_R = np.loadtxt("original_R.txt", delimiter="\t", usecols=1)
pdf_G = np.loadtxt("original_G.txt", delimiter="\t", usecols=1)
pdf_B = np.loadtxt("original_B.txt", delimiter="\t", usecols=1)
```

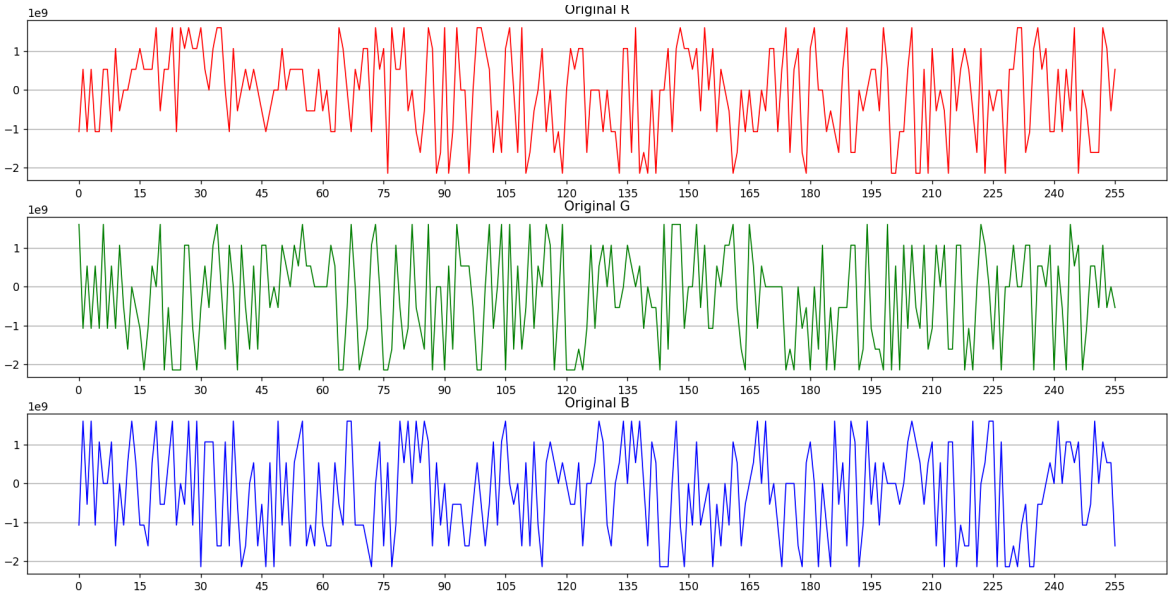
```
pdf_RO = np.loadtxt("output_R.txt", delimiter="\t", usecols=1)
pdf_GO = np.loadtxt("output_G.txt", delimiter="\t", usecols=1)
pdf_BO = np.loadtxt("output_B.txt", delimiter="\t", usecols=1)

x = np.arange(256)  # 밝기 값 0~255

fig, axs = plt.subplots(3, 1, figsize=(10, 10))

# Original R PDF
axs[0].plot(x, pdf_R, color='r', linewidth=1)
axs[0].set_title("Original R")
axs[0].set_xticks(range(0, 256, 15))
axs[0].grid(True, axis='y')

# Original G PDF
axs[1].plot(x, pdf_G, color='g', linewidth=1)
axs[1].set_title("Original G")
axs[1].set_xticks(range(0, 256, 15))
axs[1].grid(True, axis='y')

# Original B PDF
axs[2].plot(x, pdf_B, color='b', linewidth=1)
axs[2].set_title("Original B")
axs[2].set_xticks(range(0, 256, 15))
axs[2].grid(True, axis='y')

plt.tight_layout()
plt.show()



fig, axs = plt.subplots(3, 1, figsize=(10, 10))

# Output R PDF
axs[0].plot(x, pdf_RO, color='r', linewidth=1)
axs[0].set_title("Output R")
axs[0].set_xticks(range(0, 256, 15))
```

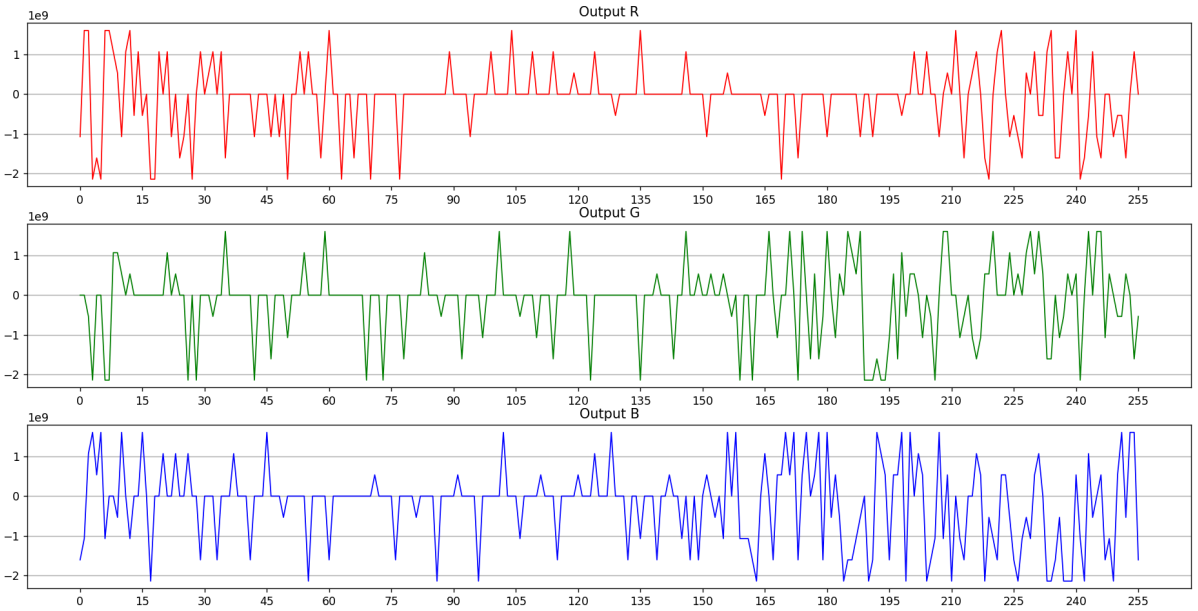
```
axs[0].grid(True, axis='y')

# Output G PDF
axs[1].plot(x, pdf_GO, color='g', linewidth=1)
axs[1].set_title("Output G")
axs[1].set_xticks(range(0, 256, 15))
axs[1].grid(True, axis='y')

# Output B PDF
axs[2].plot(x, pdf_BO, color='b', linewidth=1)
axs[2].set_title("Output B")
axs[2].set_xticks(range(0, 256, 15))
axs[2].grid(True, axis='y')

plt.tight_layout()
plt.show()
```

오리지널 이미지의 R, G, B 채널의 PDF 값과 histogram equalization을 진행한 이미지의 R, G, B 채널의 PDF 값을 plot하는 코드.

Result

plot한 결과는 다음과 같다.





위: 원본 이미지의 R, G, B의 PDF

아래: histogram equalization 된 이미지의R, G, B의 PDF

각 채널은 전반적으로 불규칙적으로 분포되어 있고, 평활화 이후 분포는 좀 더 평탄하게 변했다.

결론

차례로 input 이미지와 histogram equalization을 RGB영역에서 진행한 이미지다.

색 균형에 문제가 생겼고, 이는 곧 RGB 채널에 각각 평활화를 진행했기 때문이다.

그렇기 때문에, Y 채널을 대상으로 적용해야한다.

==========

YUV image 관련 코드 구현

hist_eq_YUV.cpp

```
1)
  Mat input = imread("input.jpg", IMREAD_COLOR);
    Mat equalized_YUV;

    cvtColor(input, equalized_YUV, COLOR_BGR2YUV);  // RGB → YUV

2)
    // split each channel(Y, U, V)
    Mat channels[3];
    split(equalized_YUV, channels);
    Mat Y = channels[0];                    // U = channels[1], V = channels[2]

3)
    float **PDF_RGB = cal_PDF_RGB(input);     // PDF of Input image(RGB) : [L][
    float *CDF_YUV = cal_CDF(Y);              // CDF of Y channel image

4)
  // histogram equalization on Y channel
    hist_eq(Y, channels[0], trans_func_eq_YUV, CDF_YUV);

5)
    // merge Y, U, V channels
    merge(channels, 3, equalized_YUV);

6)
    // YUV → RGB (use "CV_YUV2RGB" flag)
    Mat equalized_RGB;
    cvtColor(equalized_YUV, equalized_RGB, COLOR_YUV2BGR);
```

1. input 이미지를 불러오고, YUV equalization을 진행한 output 이미지를 equalized_YUV라 선언.

   RGB → YUV로 색 변경

2. Y, U, V의 각 채널들을 split

   Y 채널을 channels[0] 이라 선언.

3. input 이미지에 대해 PDF 계산, Y 채널에만 CDF 계산.

4. histogram equalization을 Y 채널에 대해 진행

    a. channels[0] == Y 채널
    b. Y를 input으로 전달
    c. channels[0]를 equalized로 전달
    d. 함수 내부에서 transfer function 적용
5. channels[0] (HE 진행)과 나머지 채널 2개를 merge해서 equalized_YUV 값 update
6. equalized_RGB를 선언해 색깔을 보기좋게 RGB로 바꿔준다

equalized_YUV_trans.py

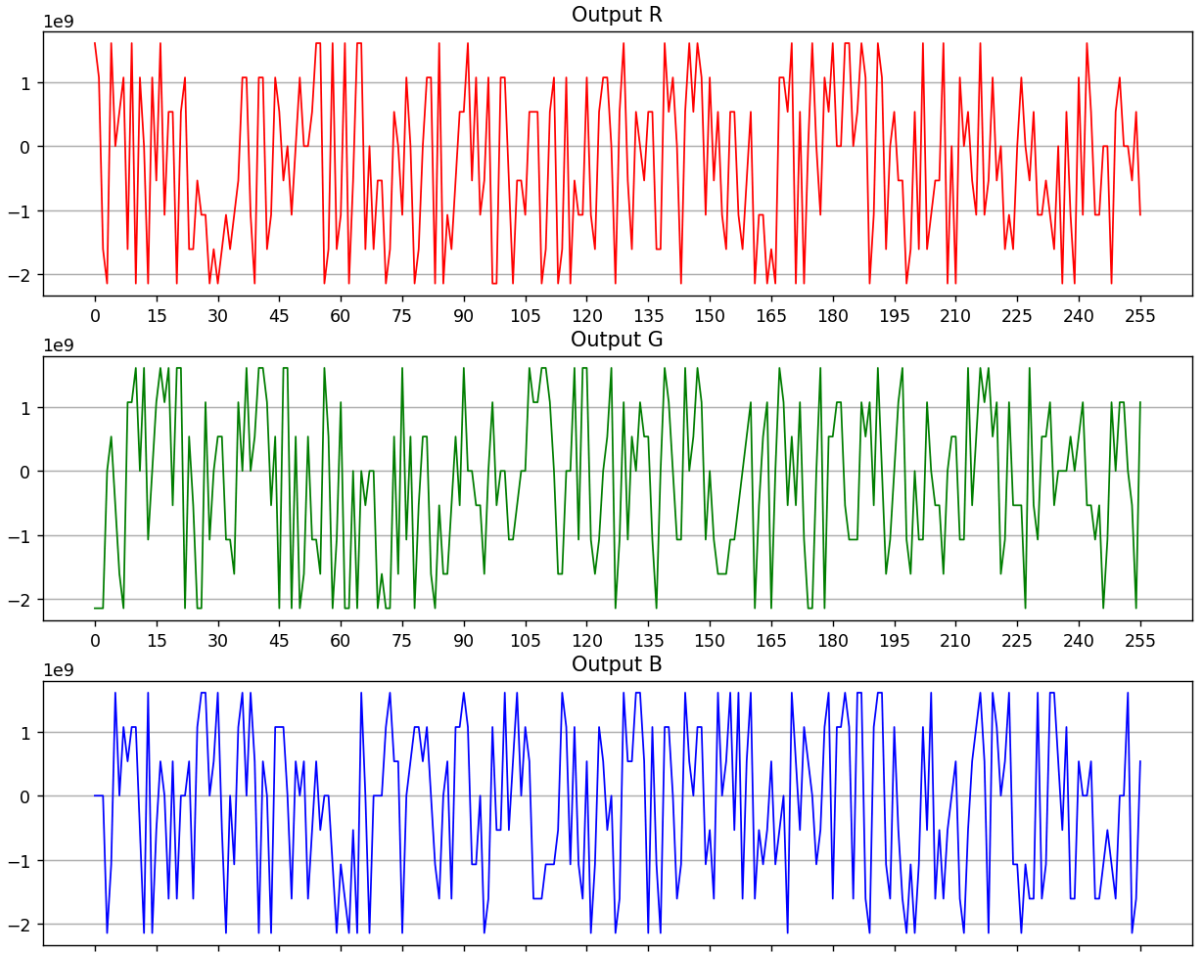
```
import matplotlib.pyplot as plt
import numpy as np

# 데이터 불러오기
pdf_R = np.loadtxt("original_R_YUV.txt", delimiter="\t", usecols=1)
pdf_G = np.loadtxt("original_G_YUV.txt", delimiter="\t", usecols=1)
pdf_B = np.loadtxt("original_B_YUV.txt", delimiter="\t", usecols=1)

pdf_RO = np.loadtxt("output_R_YUV.txt", delimiter="\t", usecols=1)
pdf_GO = np.loadtxt("output_G_YUV.txt", delimiter="\t", usecols=1)
pdf_BO = np.loadtxt("output_B_YUV.txt", delimiter="\t", usecols=1)

x = np.arange(256)  # 밝기 값 0~255

fig, axs = plt.subplots(3, 1, figsize=(10, 10))

# Original R PDF
axs[0].plot(x, pdf_R, color='r', linewidth=1)
axs[0].set_title("Original R")
axs[0].set_xticks(range(0, 256, 15))
axs[0].grid(True, axis='y')

# Original G PDF
axs[1].plot(x, pdf_G, color='g', linewidth=1)
axs[1].set_title("Original G")
axs[1].set_xticks(range(0, 256, 15))
```

```
axs[1].grid(True, axis='y')

# Original B PDF
axs[2].plot(x, pdf_B, color='b', linewidth=1)
axs[2].set_title("Original B")
axs[2].set_xticks(range(0, 256, 15))
axs[2].grid(True, axis='y')

plt.tight_layout()
plt.show()



fig, axs = plt.subplots(3, 1, figsize=(10, 10))

# Output R PDF
axs[0].plot(x, pdf_RO, color='r', linewidth=1)
axs[0].set_title("Output R")
axs[0].set_xticks(range(0, 256, 15))
axs[0].grid(True, axis='y')

# Output G PDF
axs[1].plot(x, pdf_GO, color='g', linewidth=1)
axs[1].set_title("Output G")
axs[1].set_xticks(range(0, 256, 15))
axs[1].grid(True, axis='y')

# Output B PDF
axs[2].plot(x, pdf_BO, color='b', linewidth=1)
axs[2].set_title("Output B")
axs[2].set_xticks(range(0, 256, 15))
axs[2].grid(True, axis='y')

plt.tight_layout()
plt.show()
```

오리지널 이미지의 R, G, B 채널의 PDF 값과 histogram equalization을 진행한 이미지의 R, G, B 채널의 PDF 값을 plot하는 코드.

Result

plot한 결과는 다음과 같다.



Output 이미지에 대한 PDF이다.

Y 채널 말고, RGB에 직접적으로 연산을 한 위의 hist_eq_RGB.cpp 에 비해, RGB 채널의 값이 고르게 분포되어있음을 알 수 있다.

→ Y 채널 중심의 평활화가 RGB 채널에 영향을 줬고, 색상 깨짐은 거의 없는 것으로 보임

결론

차례로 input 이미지와 histogram equalization을 Y 채널에서 진행한 이미지다.

Y 채널 기반 평활화가 색 왜곡 없이 이미지의 명암 대비를 개선했음을 보여준다.