histogram_mat_technical_report

introduction

input 이미지와 reference 이미지를 grayscale 이미지와 rgb 이미지에 대해 histogram matching을 진행한다. 그리고 오리지널 이미지에 대한 pdf과 출력 이미지에 대한 pdf을 plot 해서 분석한다.

Grayscale image 관련 코드 구현

hist_mat.cpp

main 함수

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

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

    Mat refer = imread("Equalized_YUV_image.png", IMREAD_COLOR); //refer 이미
    Mat refer_gray;

    cvtColor(refer, refer_gray, COLOR_RGB2GRAY); //gray 이미지로 변환

    Mat matched = input_gray.clone();

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]
    float* PDF_refer = cal_PDF(refer_gray); // refer 이미지의 PDF 구하기
    float* CDF_refer = cal_CDF(refer_gray); // refer 이미지의 CDF 구하기

    G match_func[L] = { 0 };  // refer 함수

3)
    hist_matching(match_func, CDF, CDF_refer);
```

```
4)
   for (int i = 0; i < input_gray.rows; i++) {
      for (int j = 0; j < input_gray.cols; j++) {
         matched.at<G>(i, j) = match_func[input_gray.at<G>(i, j)];
      }
   }
```

1. input 이미지 로드, rgb를 grayscale로 변환

   reference 이미지 로드, rgb를 grayscale로 변환

   histogram matching을 해서 출력할 이미지는 오리지널 이미지를 clone해서 생성
2. grayscale로 변환된 오리지널 이미지의 PDF, CDF 계산

   grayscale로 변환된 reference 이미지의 PDF, CDF 계산
3. histogram matching 진행

```
void hist_matching(G* match_func, float* CDF, float* CDF_refer) {
   for (int i = 0; i < L; i++) {
      float t_r = CDF[i];
      int result = 0;
      float m_diff = 1.0f;

      for (int j = 0; j < L; j++) {
         float diff = fabs(t_r - CDF_refer[j]);
         if (diff < m_diff) {
            m_diff = diff;
            result = j;
         }
      }
      match_func[i] = result;
   }
}
```

$$z = G^{-1}(s) = G^{-1}(T(r))$$

해당 공식을 사용했다.

a. input 이미지에 대해 transfer function 계산
b. refer 이미지에 대해 transfer function 계산
c. refer 이미지의 transfer function의 역함수를 구해서 원하는 output 구함

4. output으로 출력할 이미지의 픽셀값은, 위의 histogram matching 과정을 통해 계산된 픽셀값으로 채워진다.

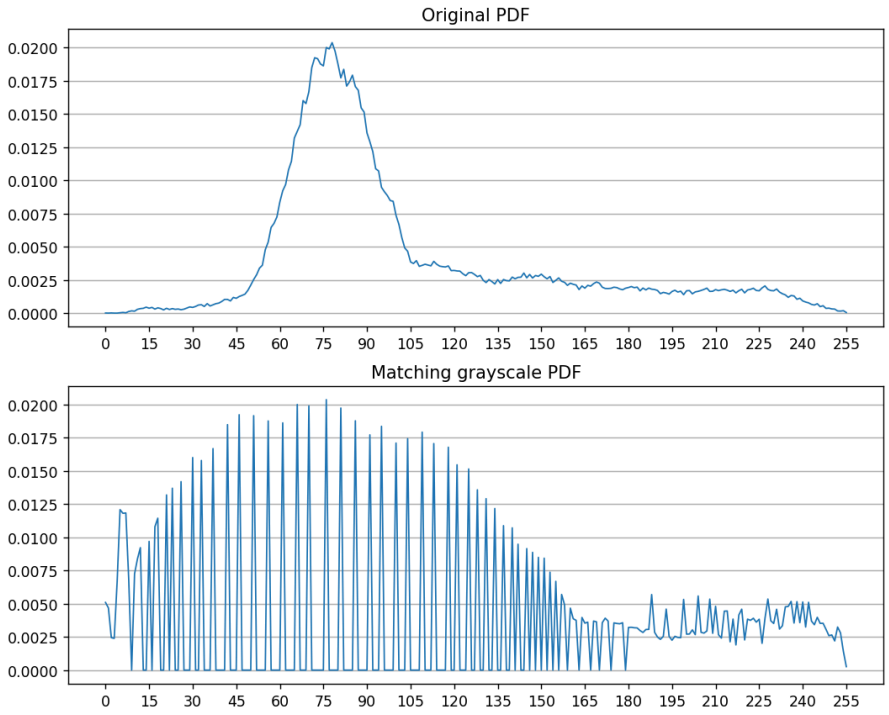
hist_matching_trans.py

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

# 데이터 불러오기
pdf = np.loadtxt("PDF.txt", delimiter="\t", usecols=1)
mat_pdf = np.loadtxt("matching_PDF_gray.txt", delimiter="\t", usecols=1)
x = np.arange(256)


fig, axs = plt.subplots(2, 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')

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

plt.show()
```

input 이미지의 PDF, histogram matching 거친 이미지의 PDF 값 plot

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 matching 된 이미지의 PDF

밝기가 70 ~ 100에 몰려 있었는데 전체적으로 균일해짐.

대비가 향상된 이미지!

매칭 대상 CDF에 맞춰 분포를 맞추다 보니, PDF 이미지가 매끄럽지 않고 뾰족뾰족한 모양이 관찰 된다.

결론

차례로 input 이미지, target 이미지, histogram matching을 진행한 이미지다.

타겟 이미지에 맞게 잘 재구성 되었으며, 이미지의 명암 대비가 향상 되었음을 알 수 있다.

==========