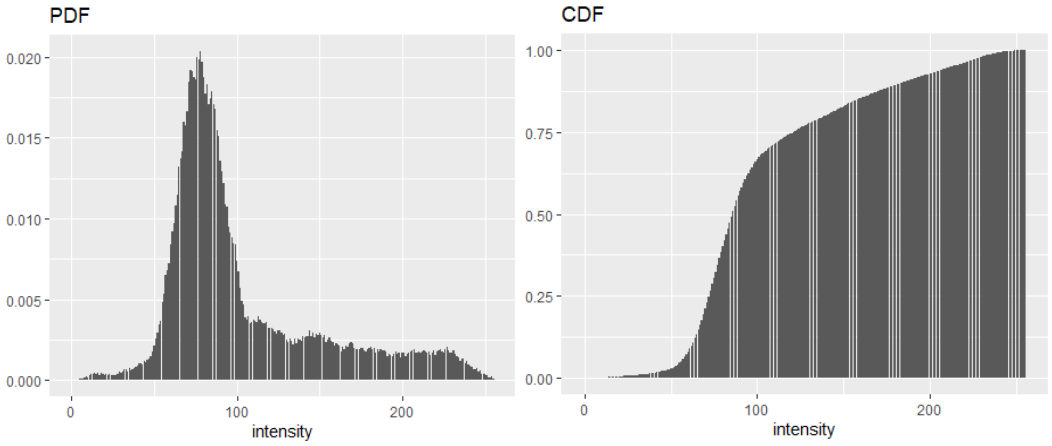
Technical Report

1829008 김민영

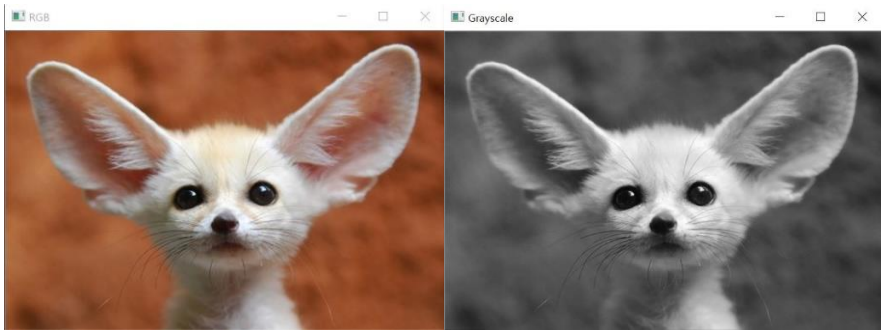
1. Practice1 – PDF/CDF Generation

Input image를 grayscale로 변환하고 출력한다. 이때, PDF와 CDF를 구해 txt 파일로 저장하고 이를 plotting 한다. (*ploting: r ggplot)



PDF는 prbabiliry density function이고 CDF는 cummulative density function이다. 이들은 아래 같은 방식으로 구현하면 된다.

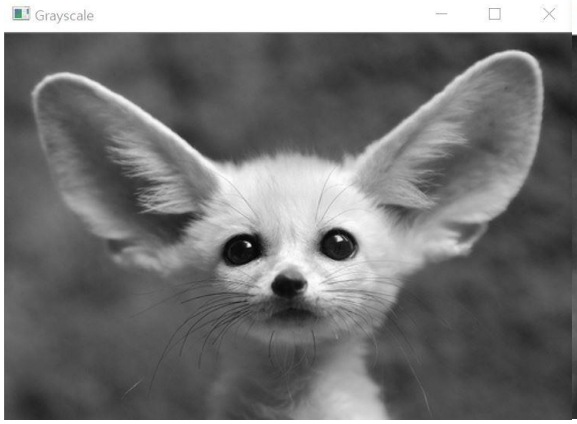
```
// Count
for (int k = 0; k < 3; k++) {
    for (int i = 0; i < input.rows; i++)
        for (int j = 0; j < input.cols; j++)
            count[input.at<C>(i,j)[k]][k] = count[input.at<C>(i, j)[k]][k] + 1;

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

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

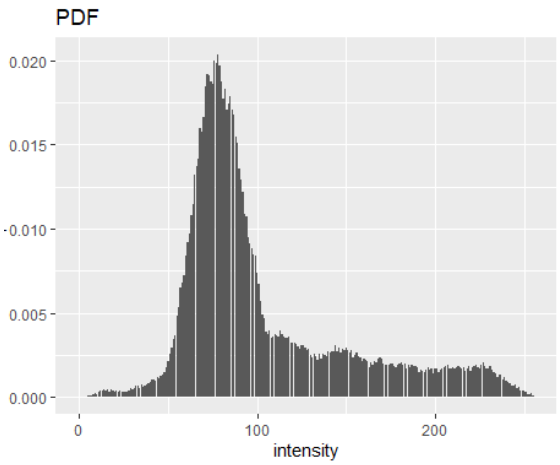


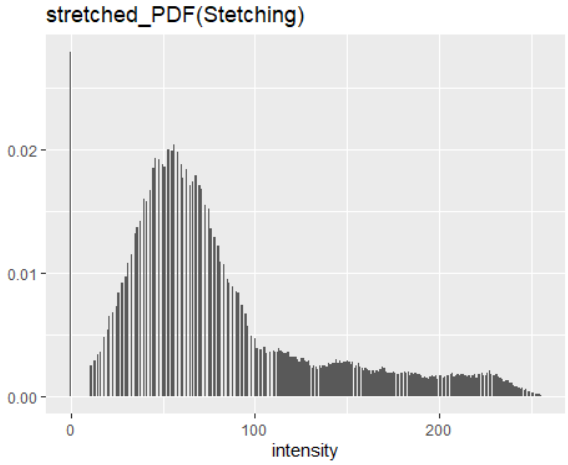
예상한 대로 잘 출력이 된 것을 확인할 수 있다.

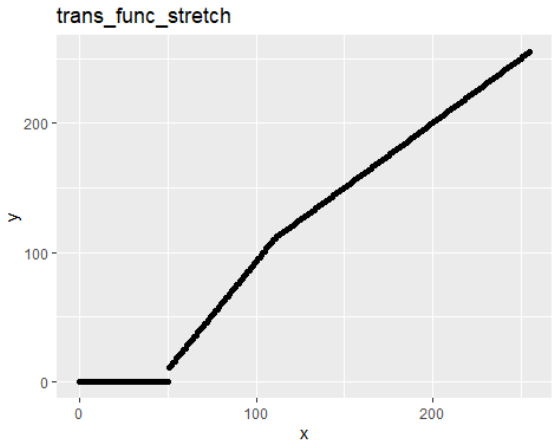
2. practice2 – Histogram Stretching

Input image를 grayscale image로 변환한다. Histogram stretching을 하고 출력한다. 이때, stretched_PDF도 저장한다.





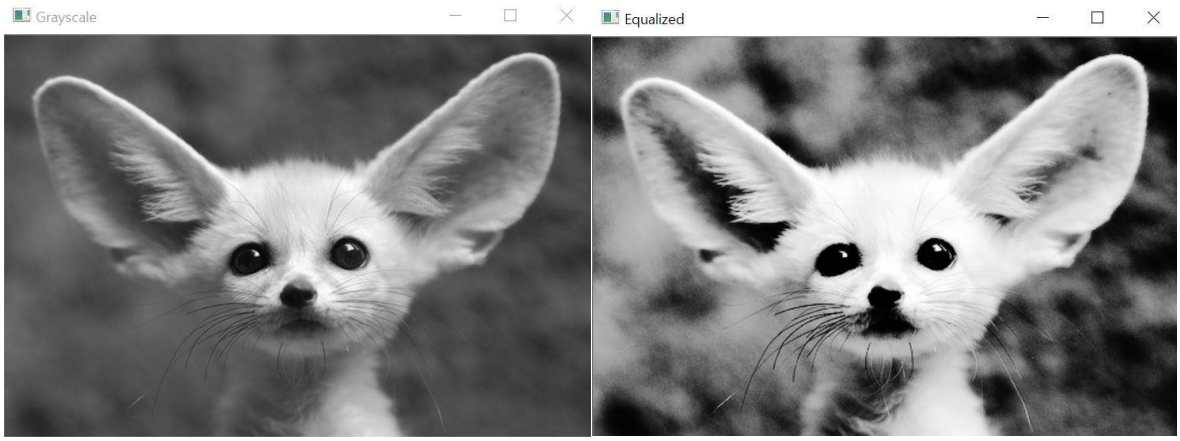


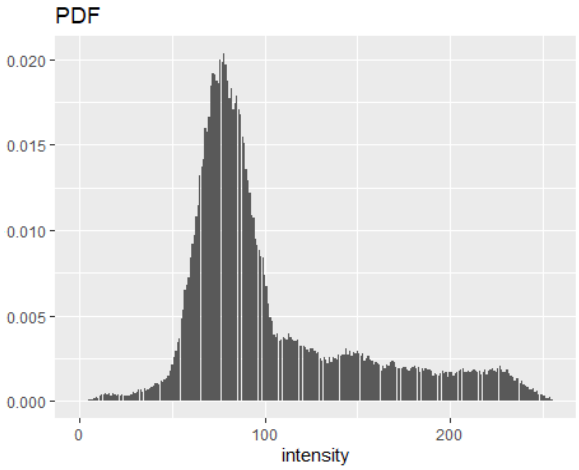


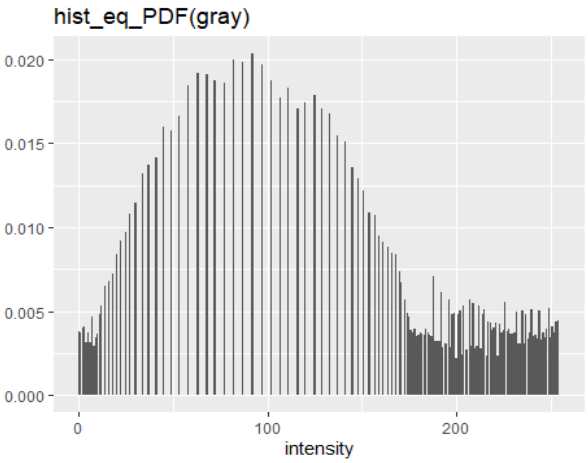
Stretched 를 하니까 PDF 가 옆으로 어느정도 퍼진 것을 확인할 수 있다. 사진또한 어느정도 대조가 된 것이 보인다.

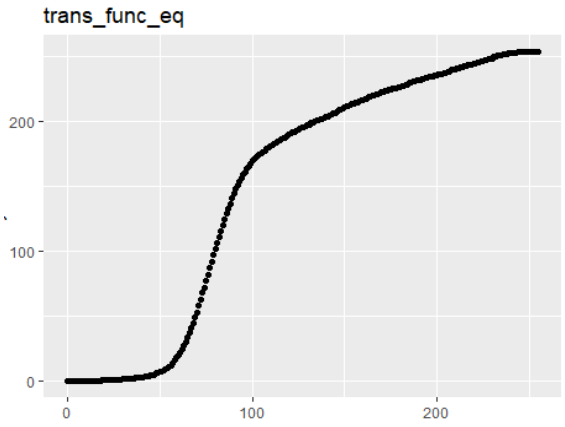
3. practice3 – Histogram Equalization Gray

Gray scale에서, Original image를 Histogram equalization 한 equalized image를 출력한다. 이때, 1 transfer function (txt file) 1 histogram of the original image (txt file) 1 histogram of the output image (txt file)를 저장한다.





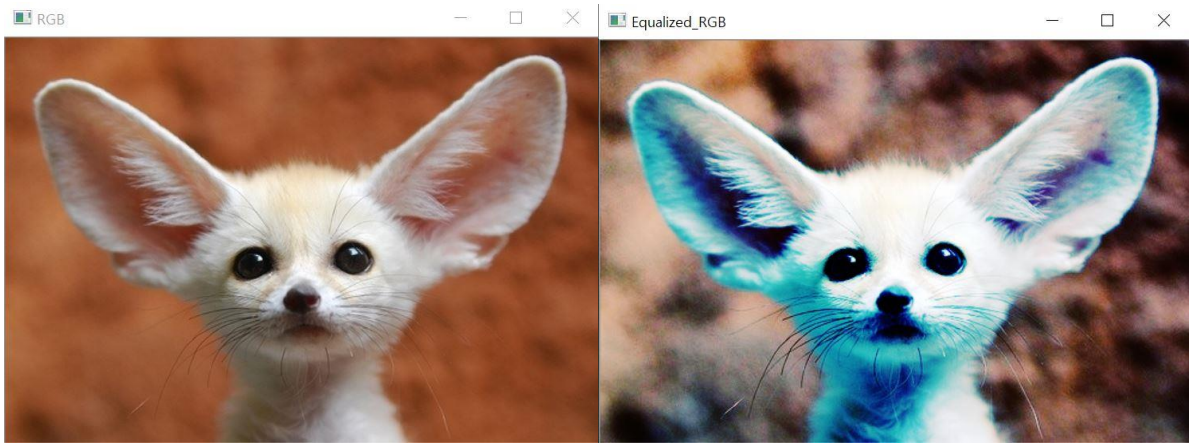


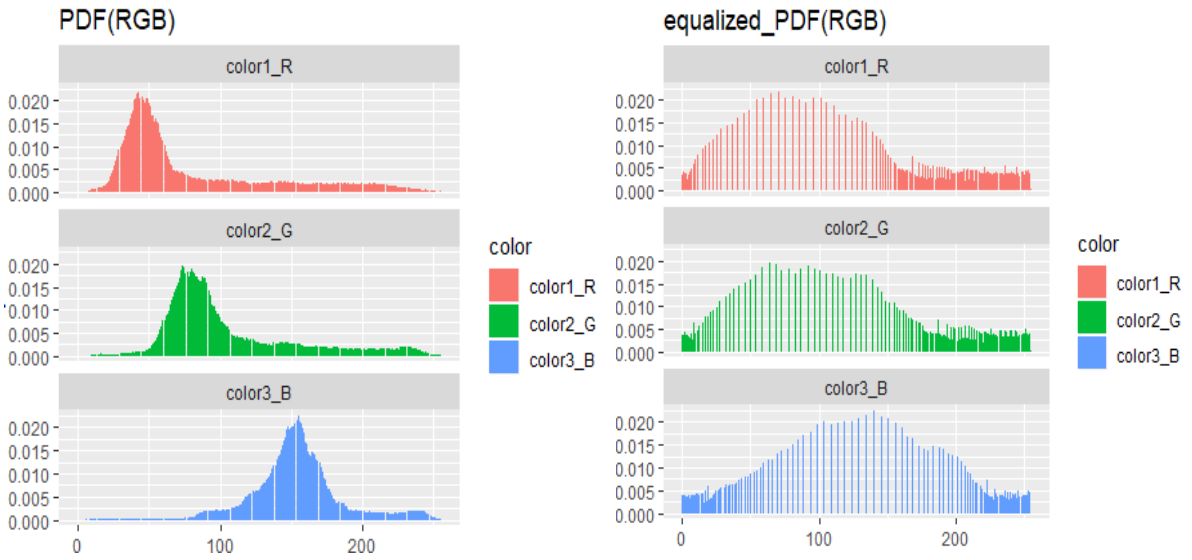


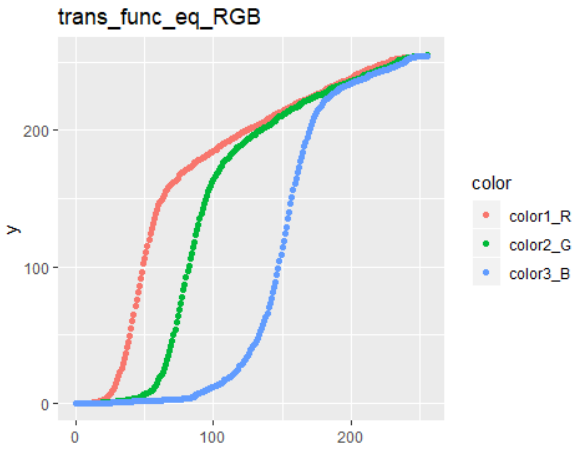
앞서 본, stretching 보다 좀더 잘 퍼져있는 것을 확인할 수 있다.

4. practice4 – Histogram Equalization RGB

RGB scale에서, Original image를 Histogram equalization 한 equalized image를 출력한다. 이때, 3 transfer functions for RGB (txt files) 3 histograms of the original imagefor RGB (txt files) 3 histograms of the output image for RGB (txt files)를 저장한다.



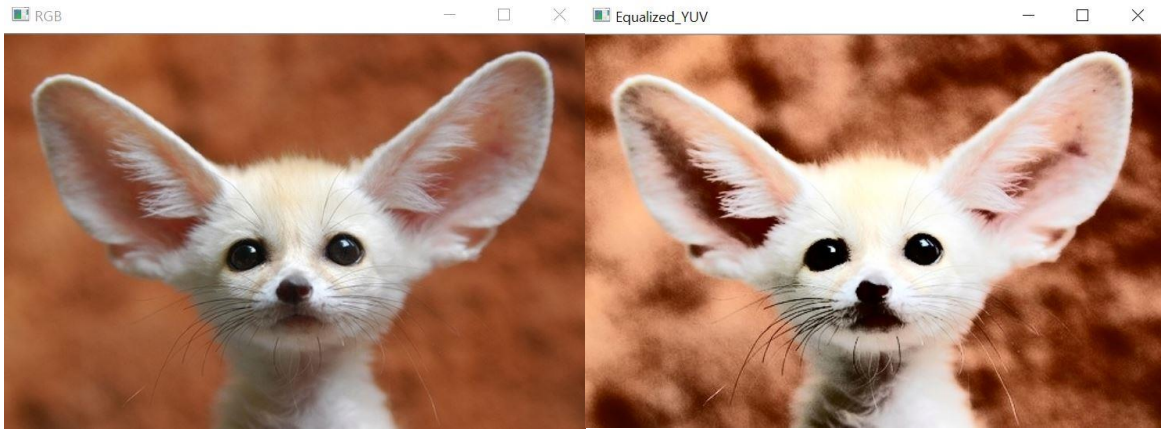


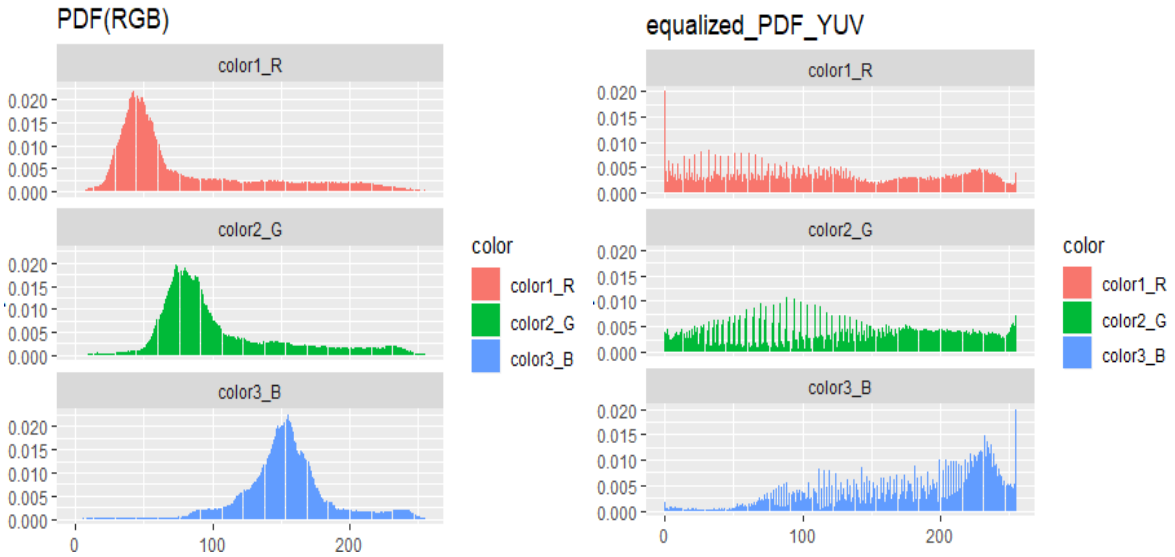


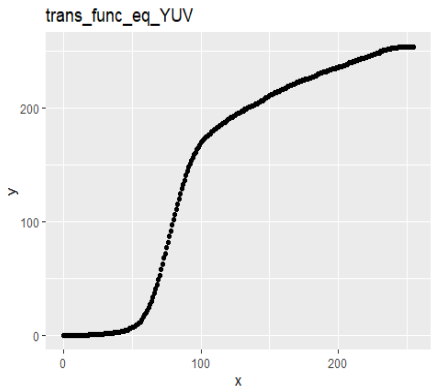
Output image 를 보면, color 가 왜곡된 것을 확인할 수 있다. RGB color 각각 하면 안된다 따라서 practice5 와 같이 문제를 해결할 수 있다.

5. practice5 – Histogram Equalization YUV

RGB color image를 YUV로 전환하고 Y channel에서만 histogram equalization을 수행한다. 이를 다시 RGB image로 전환하고 출력한다. 이때, 1 transfer functions for Y channel(txt files) 3 histograms of the original imagefor RGB (txt files) 3 histograms of the output image for RGB (txt files)를 저장한다.







앞서본 practice histogram equalization RGB 와 비교했을 때 color 가 왜곡되지 않고 잘 출력이 된 것을 확인할 수 있다.

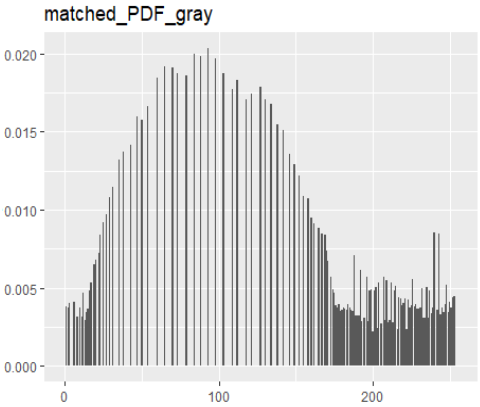
6. Homework1 – Histogram Matching Gray

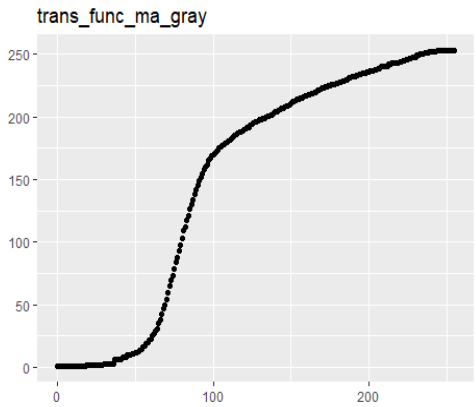
Gray scale에서, Input image와 reference image가 주어졌을 때, histogram matching을 한 matched image를 출력한다.

Hist matching 함수 : 1. compute transfer function s=T(r) 2. compute transfer function s=G(z) 3. Apply the intensity mapping from r to z z= G^(-1)(s) 와 같은 순서로 구한다.


```
#include "hist_func.h"
void hist_matching(Mat &input, Mat &matched, G *trans_func, float *CDF, float *CDF_ref);
int main() {
	// input, reference
	Mat input = imread("C:/Temp/input.jpg", CV_LOAD_IMAGE_COLOR);
	Mat input_gray;
	Mat ref = imread("C:/Temp/reference.jpg", CV_LOAD_IMAGE_COLOR);
	Mat ref_gray;
	Mat matched_gray;
	cvtColor(input, input_gray, CV_RGB2GRAY);	// convert RGB to Grayscale
	cvtColor(ref, ref_gray, CV_RGB2GRAY);		// convert RGB to Grayscale
	matched_gray = input_gray.clone();
	// PDF or transfer function txt files
	FILE *f_PDF;
	FILE *f_matched_PDF_gray;
	FILE *f_trans_func_ma_gray;
	fopen_s(&f_PDF, "PDF.txt", "w+");
	fopen_s(&f_matched_PDF_gray, "matched_PDF_gray.txt", "w+");
	fopen_s(&f_trans_func_ma_gray, "trans_func_ma_gray.txt", "w+");
	float *PDF = cal_PDF(input_gray);		// PDF of Input image(RGB) : [L][3]
	float *CDF = cal_CDF(input_gray);		// CDF of Y channel image
	float *ref_CDF = cal_CDF(ref_gray);
	G trans_func_ma_gray[L] = { 0 };		// transfer function

	// histogram matching on Y channel
	hist_matching(input_gray, matched_gray, trans_func_ma_gray, CDF, ref_CDF);
	// matched PDF (YUV)
	float *matched_PDF_gray = cal_PDF(matched_gray);
	for (int i = 0; i < L; i++) {
		// write PDF
		fprintf(f_PDF, "%d\t%f\n", i, PDF[i]);
		fprintf(f_matched_PDF_gray, "%d\t%f\n", i, matched_PDF_gray[i]);
		// write transfer functions
		fprintf(f_trans_func_ma_gray, "%d\t%d\n", i, trans_func_ma_gray[i]);
	}
	// memory release
	free(PDF);
	free(CDF);
	fclose(f_PDF);
	fclose(f_matched_PDF_gray);
	fclose(f_trans_func_ma_gray);
	////////////////////// Show each image //////////////////////
```

```
	namedWindow("input_gray", WINDOW_AUTOSIZE);
	imshow("input_gray", input_gray);
	namedWindow("reference_gray", WINDOW_AUTOSIZE);
	imshow("reference_gray", ref_gray);
	namedWindow("Matched_gray", WINDOW_AUTOSIZE);
	imshow("Matched_gray", matched_gray);
	////////////////////////////////////////////////////////////
	waitKey(0);
	return 0;
}
// histogram matching
void hist_matching(Mat &input, Mat &matched, G *trans_func, float *CDF, float *ref_CDF)
{
	// 1. compute transfer function s=T(r)   * r: input image
	G trans_func_T[L] = { 0 };
	for (int i = 0; i < L; i++)
		trans_func_T[i] = (G)((L - 1) * CDF[i]);
	// 2. compute transfer function s=G(z)   * z: reference image
	G trans_func_G[L] = { 0 };
	for (int i = 0; i < L; i++) {
		trans_func_G[i] = (G)((L - 1)*ref_CDF[i]);
	}
	// 3. Apply the intensity mapping from r to z
	// z = G^(-1)(s)
	G trans_func_G_inv[L] = { 0 };
	int s;
	for (int i = 0; i < L; i++) {
		s = trans_func_G[i];
		if (trans_func_G_inv[s] == NULL) {
			trans_func_G_inv[s] = i;
		}
	}
	s = 0;
	while (s <= L) {
		if (trans_func_G_inv[s] == NULL) {
			if (trans_func_G_inv[s - 1] == 255)
				trans_func_G_inv[s] = 255;
			else
				trans_func_G_inv[s] = trans_func_G_inv[s - 1] + 1;
		}
		s++;
	}
	for (int i = 0; i < L; i++)
		trans_func[i] = trans_func_G_inv[trans_func_T[i]];
	// perform the transfer function
	for (int i = 0; i < input.rows; i++)
		for (int j = 0; j < input.cols; j++)
			matched.at<G>(i, j) = trans_func[input.at<G>(i, j)];
}
```





➔ Input image 가 reference image 와 비슷하게 잘 매칭이 된 것을 확인할 수 있다.

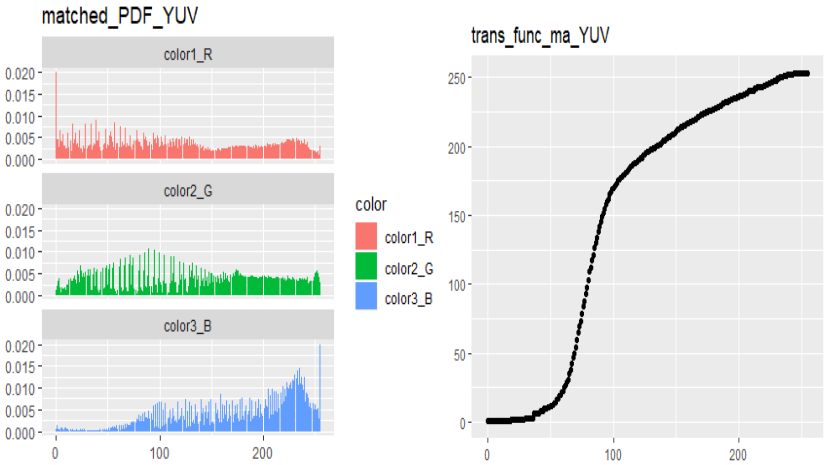
7. Homework2 – Histogram Matching YUV

Gray scale에서, Input image와 reference image가 주어졌을 때, histogram matching을 한 matced image를 출력한다. Hist_matching 부분은 앞의 homework1과 같은 방식으로 구현하였으며, main 함수에서는 추가로 1. RGB -> YUV 2. Split 3. Histogram matching 4. Merge 5. YUV->RGB 순서대로 구현하였다.


```
#include "hist_func.h"

void hist_matching(Mat &input, Mat &matched, G *trans_func, float *CDF, float *CDF_ref);
int main() {
	// input, reference
	Mat input = imread("C:/Temp/input.jpg", CV_LOAD_IMAGE_COLOR);
	Mat ref = imread("C:/Temp/reference.jpg", CV_LOAD_IMAGE_COLOR);
	Mat matched_YUV;

	// RGB -> YUV
	cvtColor(input, matched_YUV, CV_RGB2YUV);
	cvtColor(ref, ref, CV_RGB2YUV);

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

	// PDF or transfer function txt files
	FILE *f_matched_PDF_YUV, *f_PDF_RGB;
	FILE *f_trans_func_ma_YUV;
	float **PDF_RGB = cal_PDF_RGB(input);	// PDF of Input image(RGB) : [L][3]
	float *CDF_YUV = cal_CDF(Y);		// CDF of Y channel image
	float *ref_CDF_YUV = cal_CDF(ref_Y);
	fopen_s(&f_PDF_RGB, "PDF_RGB.txt", "w+");
	fopen_s(&f_matched_PDF_YUV, "matched_PDF_YUV.txt", "w+");
	fopen_s(&f_trans_func_ma_YUV, "trans_func_ma_YUV.txt", "w+");
	G trans_func_ma_YUV[L] = { 0 };		// transfer function

	// histogram matching on Y channel
	hist_matching(Y, Y, trans_func_ma_YUV, CDF_YUV, ref_CDF_YUV);

	// merge Y, U, V channels
	merge(channels, 3, matched_YUV);
```

```
	merge(ref_channels, 3, ref);

	// YUV -> RGB (use "CV_YUV2RGB" flag)
	cvtColor(matched_YUV, matched_YUV, CV_YUV2RGB);
	cvtColor(ref, ref, CV_YUV2RGB);

	// matched PDF (YUV)
	float **matched_PDF_YUV = cal_PDF_RGB(matched_YUV);
	for (int i = 0; i < L; i++) {
		// write PDF
		fprintf(f_PDF_RGB, "%d\t%f\t%f\t%f\n", i, PDF_RGB[i][0],
PDF_RGB[i][1], PDF_RGB[i][2]);
		fprintf(f_matched_PDF_YUV, "%d\t%f\t%f\t%f\n", i,
matched_PDF_YUV[i][0], matched_PDF_YUV[i][1], matched_PDF_YUV[i][2]);
		// write transfer functions
		fprintf(f_trans_func_ma_YUV, "%d\t%d\n", i, trans_func_ma_YUV[i]);
	}
	// memory release
	free(PDF_RGB);
	free(CDF_YUV);
	fclose(f_PDF_RGB);
	fclose(f_matched_PDF_YUV);
	fclose(f_trans_func_ma_YUV);
	////////////////////// Show each image //////////////////////
	namedWindow("input", WINDOW_AUTOSIZE);
	imshow("input", input);
	namedWindow("reference", WINDOW_AUTOSIZE);
	imshow("reference", ref);
	namedWindow("Matched_YUV", WINDOW_AUTOSIZE);
	imshow("Matched_YUV", matched_YUV);
	////////////////////////////////////////////////////////////
	waitKey(0);
	return 0;
}
// histogram matching
void hist_matching(Mat &input, Mat &matched, G *trans_func, float *CDF, float *ref_CDF)
{
	// 1. compute transfer function s=T(r)   * r: input image
	G trans_func_T[L] = { 0 };
	for (int i = 0; i < L; i++)
		trans_func_T[i] = (G)((L - 1) * CDF[i]);
	// 2. compute transfer function s=G(z)   * z: reference image
	G trans_func_G[L] = { 0 };
	for (int i = 0; i < L; i++) {
		trans_func_G[i] = (G)((L - 1)*ref_CDF[i]);
	}
	// 3. Apply the intensity mapping from r to z
	// z = G^(-1)(s)
	G trans_func_G_inv[L] = { 0 };
	int s;
	for (int i = 0; i < L; i++) {
		s = trans_func_G[i];
		if (trans_func_G_inv[s] == NULL) {
			trans_func_G_inv[s] = i;
		}
	}
	s = 0;
	while (s <= L) {
		if (trans_func_G_inv[s] == NULL) {
			if (trans_func_G_inv[s - 1] == 255)
				trans_func_G_inv[s] = 255;
			else
				trans_func_G_inv[s] = trans_func_G_inv[s - 1] + 1;
		}
```

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



➔ 비슷하게 잘 매칭된 것을 확인할 수 있다.