Hist_func.h

- 코드설명

```
49  float **cal_PDF_RGB(Mat &input) {
50
51      int count[L][3] = { 0 };
52      float **PDF = (float**)malloc(sizeof(float*) * L);
53
54      for (int i = 0; i < L; i++)
55          PDF[i] = (float*)calloc(3, sizeof(float));
56
57      ////////////////////////////////////////////////
58      //                                            //
59      // How to access multi channel matrix element //
60      //                                            //
61      // if matrix A is CV_8UC3 type,               //
62      // A(i, j, k) -> A.at<Vec3b>(i, j)[k]         //
63      //                                            //
64      ////////////////////////////////////////////////
65
66      // Count
67      for (int i = 0; i < input.rows; i++)
68          for (int j = 0; j < input.cols; j++)
69              for (int k = 0; k < 3; k++)
70                  count[input.at<C>(i, j)[k]][k]++;
71
72      // Compute PDF
73      for (int i = 0; i < L; i++)
74          for (int j=0; j < 3; j++)
75              PDF[i][j] = (float)count[i][j] / (float)(input.rows * input.cols);
76
77      return PDF;
78  }
```

RGB input의 PDF 값을 계산하는 함수이다. RGB의 PDF 값을 계산하기 위해서는 input의 RGB channel별로 intensity값이 몇 개씩 있는지를 알아야하기 때문에 이차원 배열 count에 값을 넣어준다. 그런 후에 count의 값들을 normalizing을 위하여 input의 row와 input의 column을 곱하여 구한 pixel 수로 나눈다.

```
102     // generate CDF for color image
103 float **cal_CDF_RGB(Mat &input) {
104
105     int count[L][3] = { 0 };
106     float **CDF = (float**)malloc(sizeof(float*) * L);
107
108     for (int i = 0; i < L; i++)
109         CDF[i] = (float*)calloc(3, sizeof(float));
110
111     ////////////////////////////////////////////////
112     //                                            //
113     // How to access multi channel matrix element //
114     //                                            //
115     // if matrix A is CV_8UC3 type,               //
116     // A(i, j, k) -> A.at<Vec3b>(i, j)[k]         //
117     //                                            //
118     ////////////////////////////////////////////////
119
120     // Count
121     for (int i = 0; i < input.rows; i++)
122         for (int j = 0; j < input.cols; j++)
123             for(int k = 0; k < 3; k++)
124                 count[input.at<C>(i, j)[k]][k]++;
```

RGB input의 CDF 값을 계산하는 함수이다. CDF는 PDF 값을 차례로 더한 값을 가지기 때문에 앞에서 구현한 cal_PDF_RGB함수와 동일하게 코딩을 하였다.

```
126     // Compute CDF
127     for (int i = 0; i < L; i++){
128         for (int j = 0; j < 3; j++) {
129             CDF[i][j] = (float)count[i][j] / (float)(input.rows * input.cols);
130 
131             if (i != 0)
132                 CDF[i][j] += CDF[i-1][j];
133         }
134     }
135 
136     return CDF;
137 }
```

그런 후에 앞의 PDF값을 차례로 더하도록 코드를 짜서 cal_CDF_RGB를 만들었다.

PDF_CDF.cpp

- 코드설명

```
16      // each histogram
17      float *PDF = cal_PDF(input_gray);       // PDF of Input image(Grayscale) : [L]
18      float *CDF = cal_CDF(input_gray);       // CDF of Input image(Grayscale) : [L]
```

Hist_func.h에서 정의한 cal_PDF와 cal_CDF 함수를 호출하여 grayscale input의 PDF, CDF 값을 계산한다.

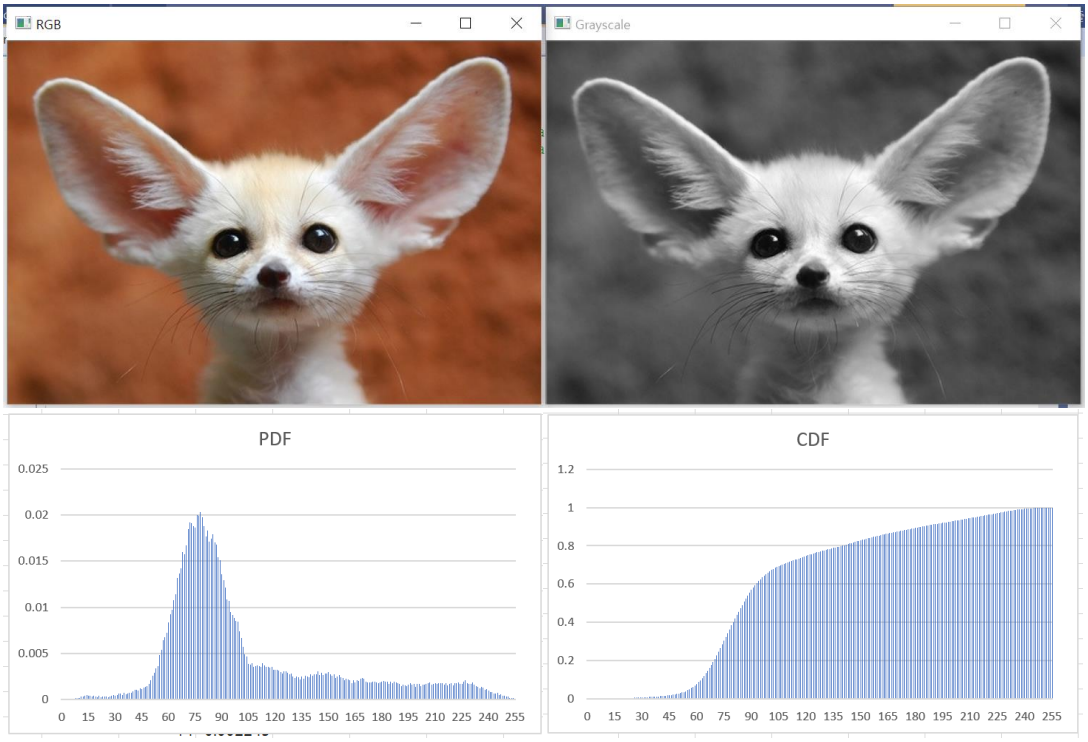
```
20      for (int i = 0; i < L; i++) {
21          // write PDF, CDF
22          fprintf(f_PDF, "%d\t%f\n", i, PDF[i]);
23          fprintf(f_CDF, "%d\t%f\n", i, CDF[i]);
24      }
```

PDF와 CDF가 각 intensity의 분포를 나타내기 때문에, intensity값을 의미하는 i를 index로 PDF와 CDF값을 각 file에 작성한다.

```
32      ////////////////////// Show each image ///////////////////////
33 
34      namedWindow("RGB", WINDOW_AUTOSIZE);
35      imshow("RGB", input);
36 
37      namedWindow("Grayscale", WINDOW_AUTOSIZE);
38      imshow("Grayscale", input_gray);
39 
40      ////////////////////////////////////////////////////////////////
```

Color image인 "input"과 input을 grayscale로 변환한 "input_gray"를 출력한다.

- 결과



- 분석

Hist_func.h에서 정의된 함수 cal_PDF, cal_cdf를 사용하여 grayscale image의 PDF와 CDF 값을 계산하는 코드이다. PDF 값은 image histogram을 pixel수로 나누어서 normalize한 값이고, CDF는 PDF 값을 cumulative하게 더한 값이다.

hist_stretching.cpp

- 코드설명

```
27    linear_stretching(input_gray, stretched, trans_func_stretch, 50, 110, 10, 110); // histogram stretching (x1 ~ x2 -> y1 ~ y2)
```

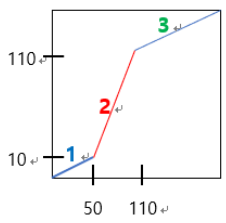
Histogram stretching을 해주는 함수이다.

"input_gray" : Histogram stretching하고자 하는 input.

"stretched" : histogram stretched image를 넣기 위해 같은 사이즈의 "input_gray"로 초기화한 Mat class.

"trans_func_stretched" : histogram stretching의 transfer function이 들어갈 빈 G형 배열. 0으로 초기화.

이 함수에 input을 넣으면



위 그래프에 따라서 intensity값이 0~50은 1번 그래프, 50~110은 2번 그래프, 110~255는 3번 그래프를 따라서 intensity 값이 변하게 된다.

```
64      float constant = (y2 - y1) / (float)(x2 - x1);
```

50~110 intensity를 stretch할 linear의 기울기를 계산한다.

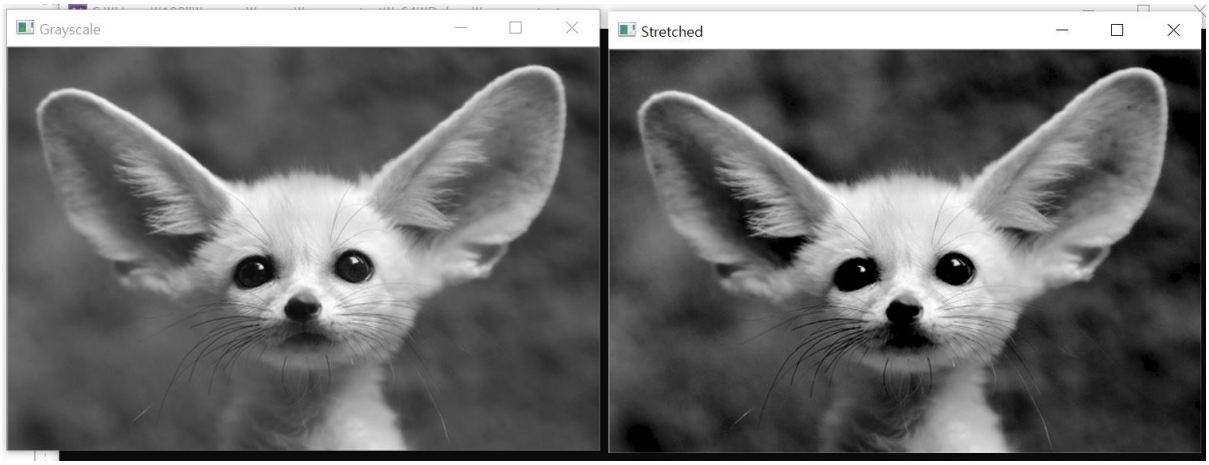
```
66      // compute transfer function
67      for (int i = 0; i < L; i++) {
68          if (i >= 0 && i <= x1)
69              trans_func[i] = (G)(y1 / x1 * i);
70          else if (i > x1 && i <= x2)
71              trans_func[i] = (G)(constant * (i - x1) + y1);
72          else
73              trans_func[i] = (G)((L - 1 - x2) / (L - 1 - y2) * (i - x2) + y2);
74      }
```

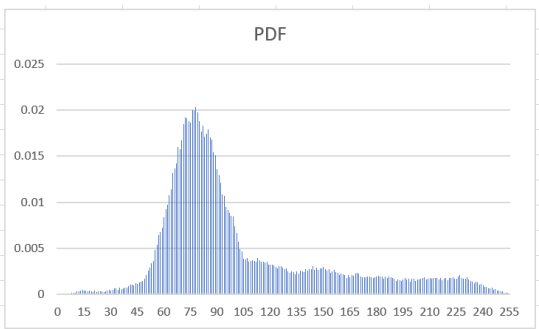
첫번째 조건문에는 0~50 intensity, 두번째 조건문에는 50~110 intensity, 세번째 조건문에는 110~255 intensity의 transfer function이 만들어진다.

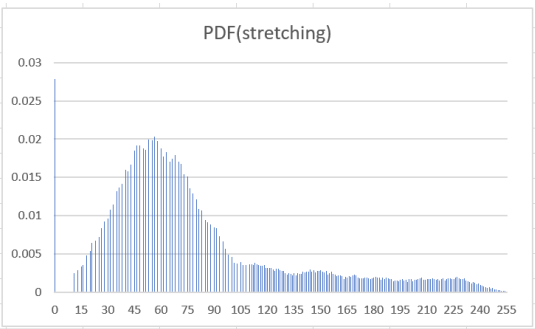
```
76      // perform the transfer function
77      for (int i = 0; i < input.rows; i++)
78          for (int j = 0; j < input.cols; j++)
79              stretched.at<G>(i, j) = trans_func[input.at<G>(i, j)];
```

trans_func은 intensity 값을 index로 하기 때문에 input의 각 pixel에서의 intensity 값으로 접근할 수 있다. 이렇게 위에서 만든 trans_func을 이용하여 histogram stretching된 image, "stretched"를 만든다.

- 결과







- 분석

이 코드는 grayscale input의 histogram stretching을 위한 코드이다. 보통 input의 contrast를 높이기 위하여 histogram stretching을 한다. Contrast가 높아지면 더 나은 visibility를 얻을 수 있다. Input의 pdf와 output의 pdf를 비교해보면 코드에서 설정한대로 50~110 intensity가 10~110으로 stretch되어서 분포하는 것을 확인할 수 있다.

hist_eq.cpp

- 코드설명

```
28      hist_eq(input_gray, equalized, trans_func_eq, CDF);
```

"input_gray" : Grayscale input에 대한 equalization이므로 gray로 변환한 image를 input으로 사용.
"equalized" : histogram equalization된 이미지를 받기 위해 같은 사이즈의 "input_gray"로 초기화한 Mat class.
"trans_func_eq" : histogram equalization의 transfer function이 들어갈 빈 G형 배열. 0으로 초기화.
"CDF" : histogram equalization을 계산할 때 필요한 grayscale input의 CDF 값.

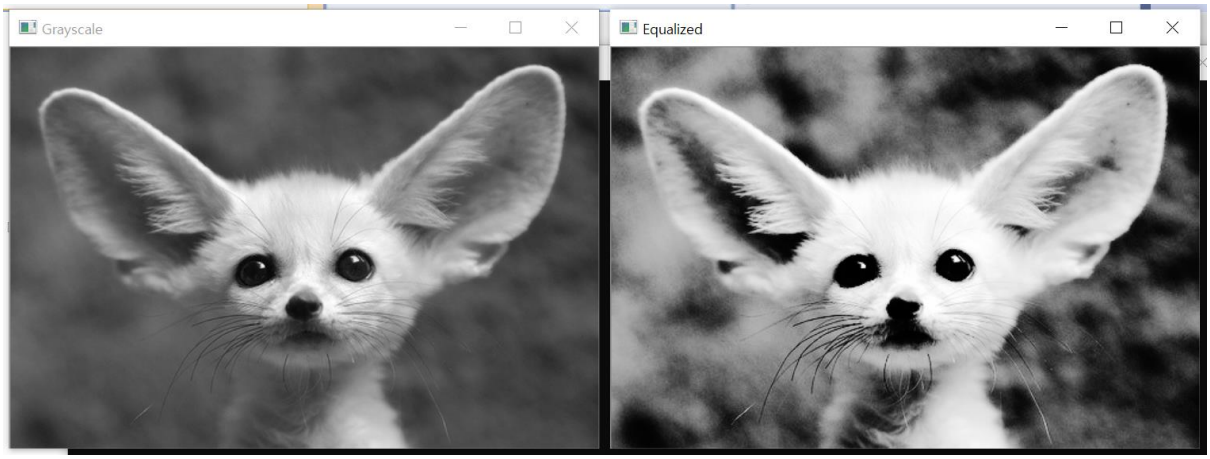
```
65      // compute transfer function
66      for (int i = 0; i < L; i++)
67          trans_func[i] = (G)((L - 1) * CDF[i]);
```

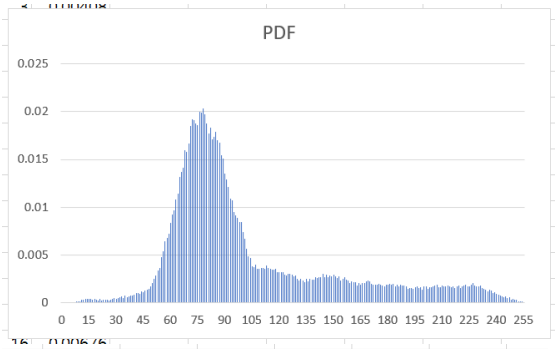
Histogram equalization을 계산하기 위한 식은 output=(L-1)*CDF 이다. 이 식을 "trans_func"에 넣어준다.

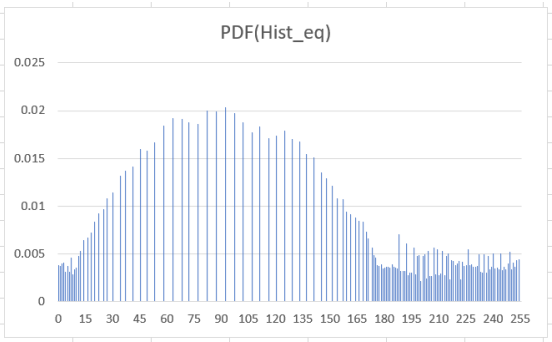
```
69      // perform the transfer function
70      for (int i = 0; i < input.rows; i++)
71          for (int j = 0; j < input.cols; j++)
72              equalized.at<G>(i, j) = trans_func[input.at<G>(i, j)];
73  }
```

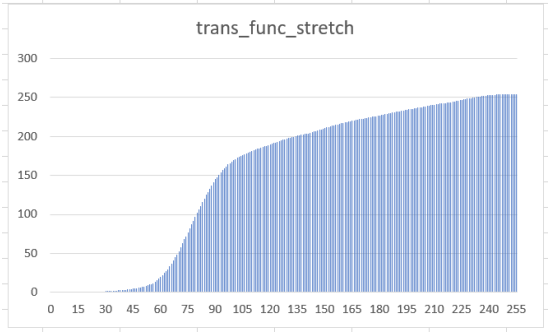
"trans_func"은 intensity를 index로 하고 있기 때문에 input의 각 픽셀이 가지는 intensity 값으로 접근할 수 있고, 위에 코드에서 만든 "trans_func"을 이용하여 input의 픽셀들을 histogram equalization해서 "equalized"변수에 넣어준다.

- 결과









- 분석

Image equalization 또한 histogram stretching과 같이 image의 contrast를 높일 때 사용한다. 이 코드는 grayscale input의 histogram equalization을 위한 것이다. output=(L-1)*CDF 식을 이용하여 histogram equalization을 한 후에 input PDF의 histogram과 output PDF의 histogram을 비교해보면 output PDF histogram의 모양이 input PDF histogram 보다 uniform해진 것을 파악할 수 있다.

hist_eq_RGB

- 코드설명

```
23      // histogram equalization on RGB image
24      hist_eq_Color(input, equalized_RGB, trans_func_eq_RGB, CDF_RGB);
```

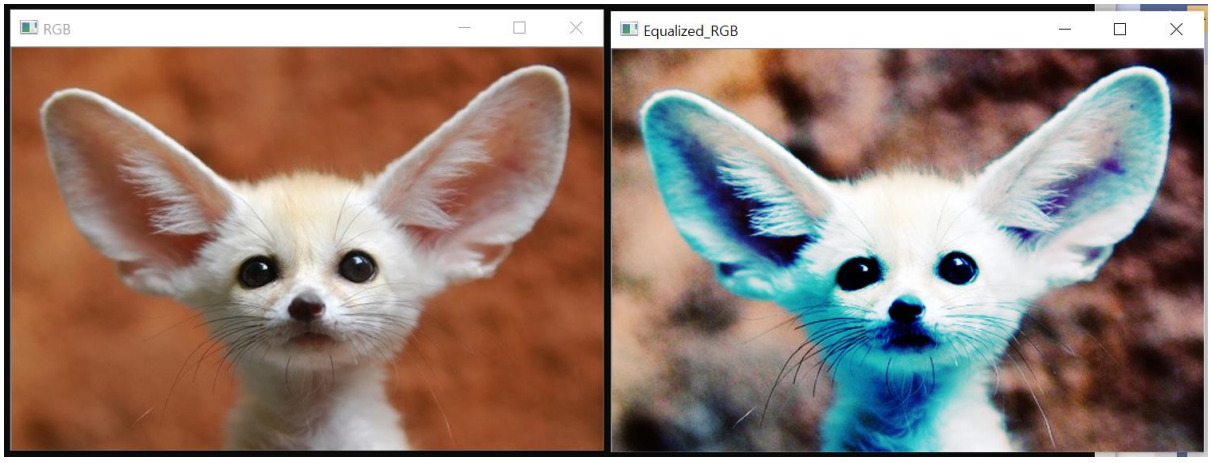
Hist_Eq_Color는 color input을 histogram equalization 해주는 함수이다.

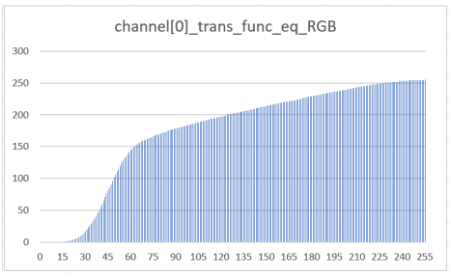
"input" : histogram equalization을 적용할 color image.
"equalized_RGB" : histogram equalization된 이미지를 받기 위해 같은 사이즈의 "input"로 초기화한 Mat class.
"trans_func_Eq_RGB" : histogram equalization의 transfer function이 들어갈 이차원 G형 배열. 0으로 초기화.
"CDF_RGB" : histogram equalization을 계산할 때 필요한 color input의 CDF 값.

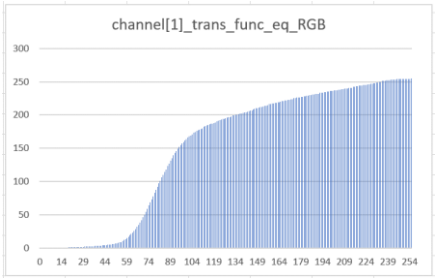
```
62    // histogram equalization on 3 channel image
63    void hist_eq_Color(Mat &input, Mat &equalized, G(*trans_func)[3], float **CDF) {
64
65        ////////////////////////////////////////////////
66        //                                            //
67        // How to access multi channel matrix element //
68        //                                            //
69        // if matrix A is CV_8UC3 type,               //
70        // A(i, j, k) -> A.at<Vec3b>(i, j)[k]         //
71        //                                            //
72        ////////////////////////////////////////////////
73
74        // compute transfer function
75        for (int i = 0; i < L; i++)
76            for (int j = 0; j < 3; j++)
77                trans_func[i][j] = (G)((L - 1) * CDF[i][j]);
78
79        // perform the transfer function
80        for (int i = 0; i < input.rows; i++)
81            for (int j = 0; j < input.cols; j++)
82                for (int k=0;k<3;k++)
83                    equalized.at<C>(i, j)[k] = trans_func[input.at<C>(i, j)[k]][k];
84    }
```

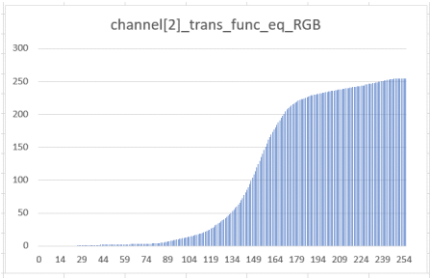
RGB 각각에 대하여 transfer function을 만들기 위하여 CDF 이차원 배열에 j를 이용하여 접근하도록 했다.
"trans_func"은 [intensity value][channel num]으로 index가 있어서 우선 i와 j를 이용해서 "input"의 픽셀에 접근하였고 그때의 intensity값을 "trans_func"의 index로 사용하였다. 각 RGB channel별로 "trans_func"을 적용하여 histogram equalization된 image를 "equalized"에 넣도록 "hist_eq_Color" 함수를 만들었다.

- 결과

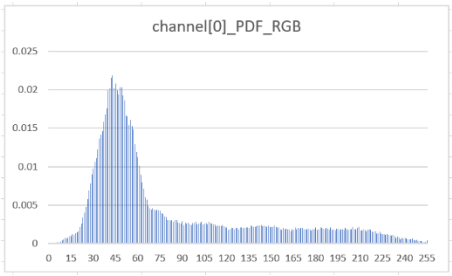


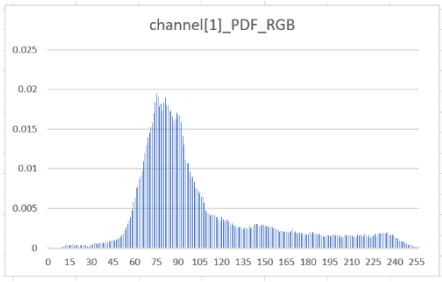


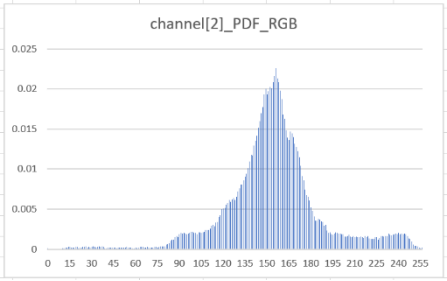




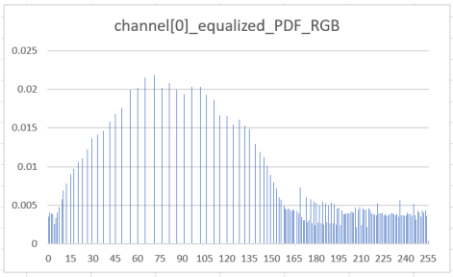
transfer functions for RGB

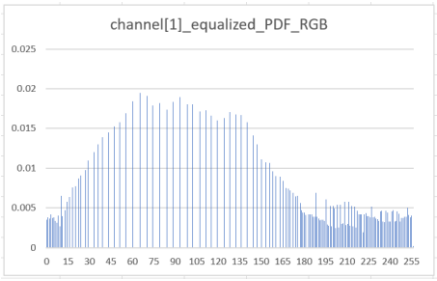


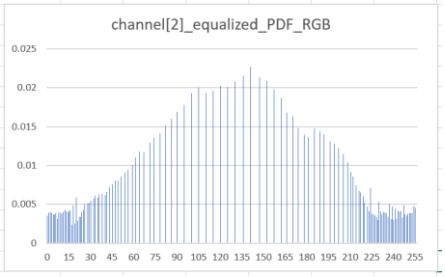




histograms of the original image for RGB







Histograms of the output image for RGB

- 분석

Histogram equalization은 image contrast를 높여서 visibility를 더 좋게 할 수 있다. 이 코드는 RGB input을 histogram equalization으로 RGB channel을 각각 나누어서 계산한다. 사실 color input의 경우에는 YUV로 변환하여서 Y channel에만 histogram equalization을 해야 output에 color distortion이 발생하지 않는다. histogram equalization을 한 후에 input PDF의 histogram과 output PDF의 histogram을 비교해보면 output PDF histogram의 모양이 input PDF histogram 보다 uniform해진 것을 파악할 수 있다.

hist_eq_YUV

- 코드설명

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

Input이 아닌 input의 Y channel에만 histogram equalization을 적용하기 때문에 Y에 대해서만 CDF 값을 계산하여 사용한다. 또한 여러 channel이 있는 input이 아니기 때문에 cal_CDF_RGB 함수가 아닌 cal_CDF 함수를 이용하여 CDF값을 얻는다.

```
30      // histogram equalization on Y channel
31      hist_eq(Y, Y, trans_func_eq_YUV, CDF_YUV);
```

Histogram equalization함수.

"Y" : histogram equalization을 적용할 input.

"Y" : histogram equalization이 적용된 output을 받아 옴.

"trans_func_eq_YUV" : histogram equalization의 transfer function이 들어갈 G형 배열. 0으로 초기화.

"CDF_YUV" : histogram equalization을 계산할 때 필요한 "Y"의 CDF 값.

```
33      // merge Y, U, V channels
34      merge(channels, 3, equalized_YUV);
```

Histogram equalization된 값이 반영된 channels를 다시 합쳐서 equalized_YUV에 넣어준다.

```
36      // YUV -> RGB (use "CV_YUV2RGB" flag)
37      cvtColor(equalized_YUV, equalized_YUV, CV_YUV2RGB);
```

YUV로 변환했던 이미지를 RGB로 다시 변환한다.

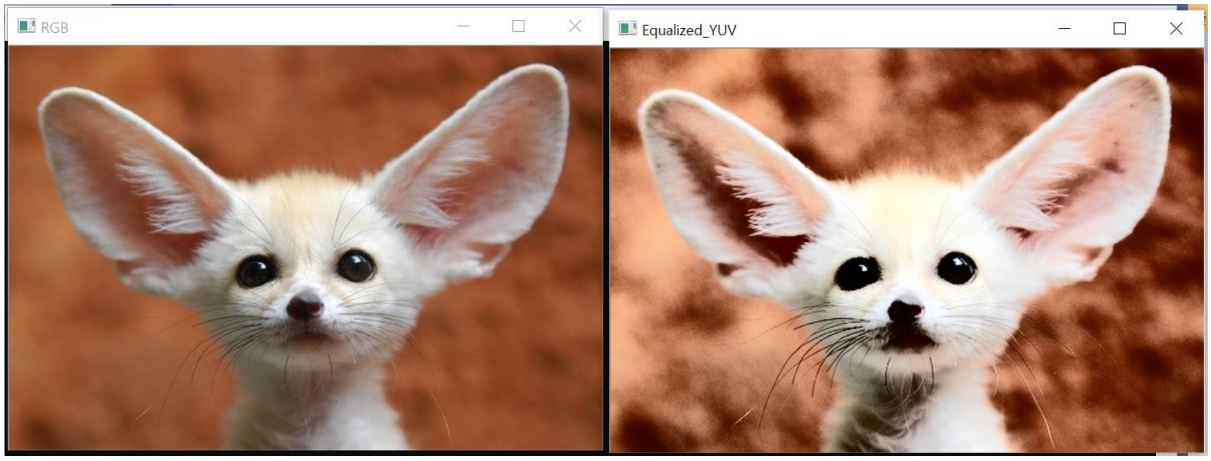
```
39      // equalized PDF (YUV)
40      float** equalized_PDF_YUV = cal_PDF_RGB(equalized_YUV);
```

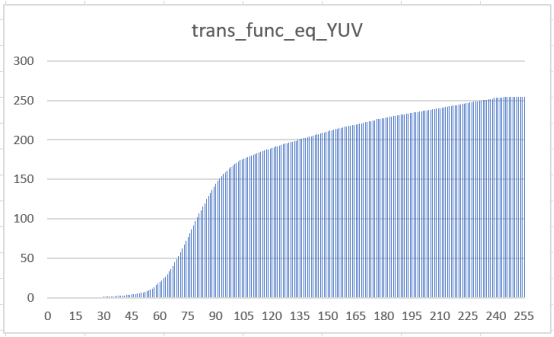
Equalized_YUV가 RGB이미지이므로 이중 포인터를 사용하였다.

```
42      for (int i = 0; i < L; i++) {
43          for (int j = 0; j < 3; j++) {
44              // write PDF
45              fprintf(f_PDF_RGB, "%d₩t%d₩t%f₩n", i, j, PDF_RGB[i][j]);
46              fprintf(f_equalized_PDF_YUV, "%d₩t%d₩t%f₩n", i, j, equalized_PDF_YUV[i][j]);
47          }
48
49          // write transfer functions
50          fprintf(f_trans_func_eq_YUV, "%d₩t%d₩n", i, trans_func_eq_YUV[i]);
51      }
```

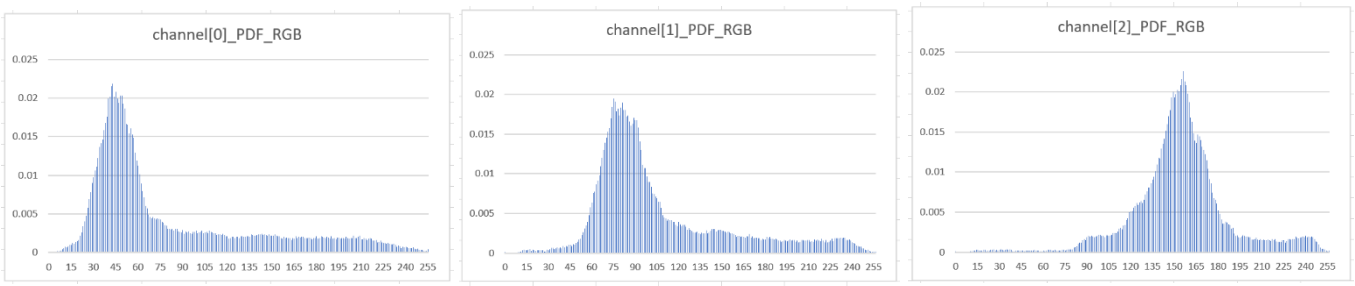
File에 output을 기록하기 위하여 위의 코드를 작성한다. PDF_RGB, equalized_PDF_YUV의 경우 3개의 channel이 있고, 다른 값을 가지고 있기 때문에 각 RGB channel에 접근할 index가 필요하다. 따라서 이 두개의 값은 i와 j를 두개를 index로 사용하였다. trans_func_eq_YUV는 Y channel 하나에 대한 transfer function이기 때문에 하나의 index만으로 접근할 수 있어서 i 하나를 index로 사용하였다.

- 결과

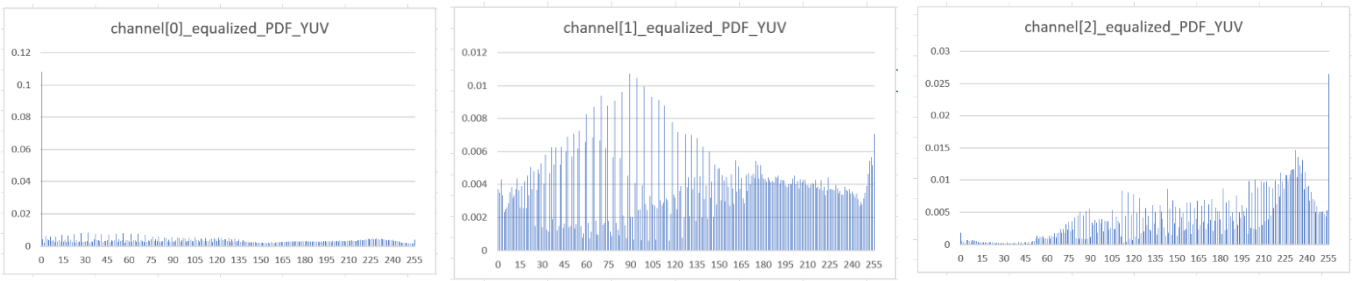




Transfer function for Y channel



Histograms of the original image for RGB



histograms of the output image for RGB

- 분석

Histogram equalization은 image contrast를 높여서 visibility를 더 좋게 할 수 있다. Color input을

histogram equalization하는 경우에는 RGB를 YUV로 변환한 수에 Y channel에만 transfer를 해야 한다. 그렇지 않으면 color distortion이 발생한다.

hist_matching.cpp

- 코드설명

```
8     Mat input = imread("input.jpg", CV_LOAD_IMAGE_COLOR);
9     Mat ref = imread("ref.jpg", CV_LOAD_IMAGE_COLOR);
10    Mat input_gray;
11    Mat ref_gray;
12
13    cvtColor(input, input_gray, CV_RGB2GRAY);   // convert RGB to Grayscale
14    cvtColor(ref, ref_gray, CV_RGB2GRAY);   // convert RGB to Grayscale
```

Grayscale input을 histogram matching하는 코드이므로 Input과 reference를 읽어서 gray color로 변환한다.

```
19    // PDF or transfer function txt files
20    FILE* f_PDF;
21    FILE* f_hist_matching_PDF;
22    FILE* f_trans_func_matching;
23
24    fopen_s(&f_PDF, "PDF.txt", "w+");
25    fopen_s(&f_hist_matching_PDF, "hist_matching_PDF.txt", "w+");
26    fopen_s(&f_trans_func_matching, "trans_func_matching.txt", "w+");
```

Input의 PDF값, histogram matching한 결과의 PDF값, 이때의 transfer function 값을 txt 파일에 저장하기 위하여 변수를 선언하고 파일을 만든다.

```
35    hist_eq(input_gray, equalized, trans_func_eq, CDF);
```

"input_gray" : Grayscale input에 대한 equalization이므로 gray로 변환한 image를 input으로 사용.
"equalized" : histogram equalization된 이미지를 받기 위해 같은 사이즈의 "input_gray"로 초기화한 Mat class.
"trans_func_eq" : histogram equalization의 transfer function이 들어갈 빈 G형 배열. 0으로 초기화.
"CDF" : histogram equalization을 계산할 때 필요한 grayscale input의 CDF 값.

```
42    hist_eq(ref_gray, equalized_ref, trans_func_eq_ref, CDF_ref);
```

"ref_gray" : reference를 gray로 변환한 이미지 input으로 사용.
"equalized_ref" : histogram equalization된 이미지를 받기 위해 같은 사이즈의 "ref_gray"로 초기화한 Mat class.
"trans_func_eq_ref" : histogram equalization의 transfer function이 들어갈 빈 G형 배열. 0으로 초기화.
"CDF_ref" : histogram equalization을 계산할 때 필요한 grayscale reference의 CDF 값.

```
44    hist_matching(input_gray, matched, trans_func_eq, trans_func_eq_ref, trans_func_matching);
```

"input_gray" : histogram matching을 할 input을 넣는다.
"matched" : histogram matching된 결과를 받기 위해 같은 사이즈의 "input_gray"로 초기화한 Mat

class.

“trans_func_eq” : “input_gray”를 histogram equalization 할 때의 transfer function.

“trans_func_eq_ref” : “ref_gray”를 histogram equalization할 때의 transfer function.

“trans_func_matching” : histogram matching의 transfer function이 들어갈 빈 G형 배열. 0으로 초기화.

```
81      // histogram equalization
82      void hist_eq(Mat& input, Mat& equalized, G* trans_func, float* CDF) {
83
84          // compute transfer function
85          for (int i = 0; i < L; i++)
86              trans_func[i] = (G)((L - 1) * CDF[i]);
87
88          // perform the transfer function
89          for (int i = 0; i < input.rows; i++)
90              for (int j = 0; j < input.cols; j++)
91                  equalized.at<G>(i, j) = trans_func[input.at<G>(i, j)];
92      }
```

Histogram equalization을 계산하기 위한 식은 output=(L-1)*CDF 이다. 이 식을 “trans_func”에 넣어준다. “trans_func”은 intensity를 index로 하고 있기 때문에 input의 각 픽셀이 가지는 intensity 값으로 접근할 수 있고, 위에 코드에서 만든 “trans_func”을 이용하여 input의 픽셀들을 histogram equalization해서 “equalized”변수에 넣어준다.

```
94      // histogram matching
95      void hist_matching(Mat& input, Mat& matched, G* trans_func_eq, G* trans_func_ref, G*trans_func_matching) {
96          G temp[L] = { 0 };
97
98          //compute transfer function
99          for (int i = 0; i < L; i++) {
100             for (int j = 0; j < L; j++) {
101                 if (i == trans_func_ref[j]) {
102                     temp[i] = j;
103                     break;
104                 }
```

Trans_func_ref의 역함수를 구하는 코드이다. Y=f(x) 함수를 생각했을 때 trans_func_ref의 index가 x이고, 그 위치에 있는 값은 y이다. 따라서 여기서 역함수를 만들기 위해서는 x와 y의 위치를 바꿔서 x=f(y)로 만들어야 하므로 temp를 이용하여 바꿔준다.

```
105                 if (j == L - 1)
106                     temp[i] = temp[i - 1] + 1;
```

하지만 역함수 x=f(y)에서 x값을 갖지 못하는 y가 발생하면 함수가 아니므로 이를 방지해야 한다. 따라서 만약 그러한 y가 있는 경우 이전 y가 가진 x값에 +1을 한 값을 주도록 했다.

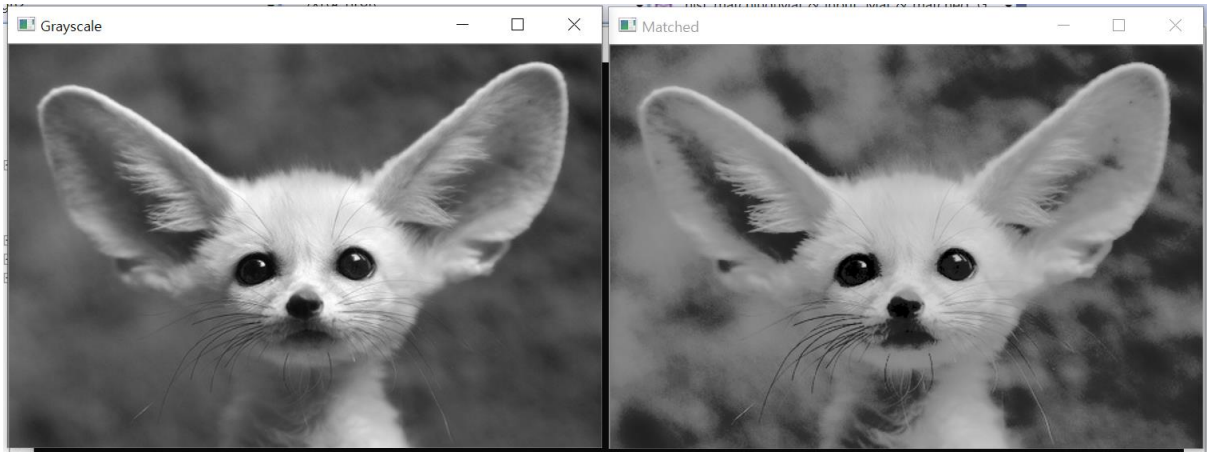
```
110         for (int i = 0; i < L; i++)
111             trans_func_matching[i] = temp[trans_func_eq[i]];
```

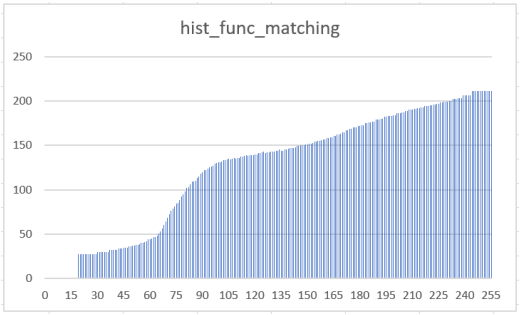
위에서 구한 역함수 temp와 trans_func_eq를 합성하여서 histogram matching을 위한 transfer function, trans_func_matching을 만든다.

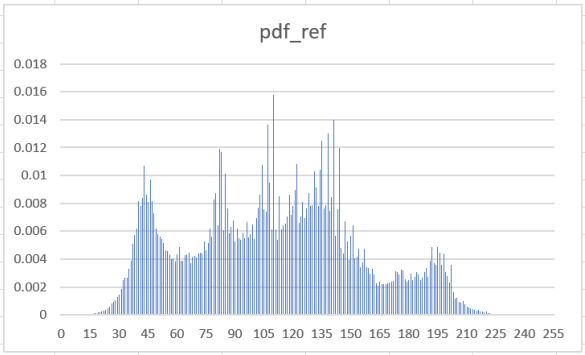
```
113         //perform the transfer function
114         for (int i = 0; i < input.rows; i++)
115             for (int j = 0; j < input.cols; j++)
116                 matched.at<G>(i, j) = trans_func_matching[input.at<G>(i, j)];
```

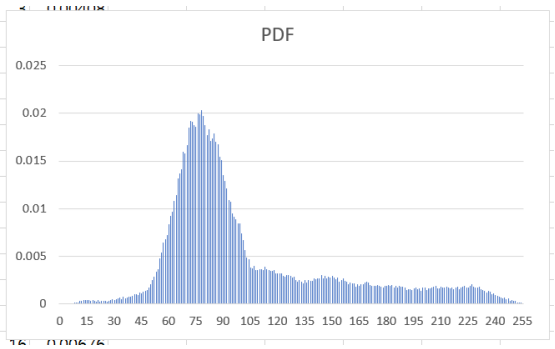
."trans_func_matching"은 intensity를 index로 하고 있기 때문에 input의 각 픽셀이 가지는 intensity 값으로 접근할 수 있고, 위에 코드에서 만든 "trans_func_matching"을 이용하여 input의 픽셀들을 histogram matching해서 "matched"변수에 넣어준다.

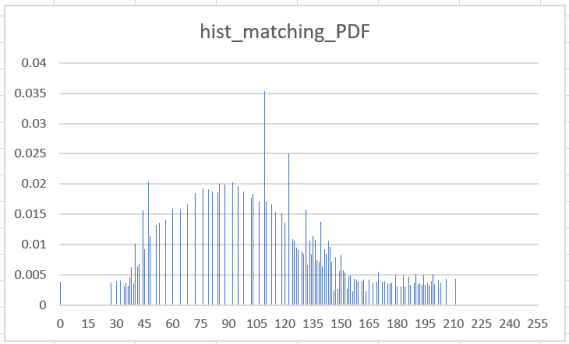
- 결과











- 분석

histogram matching을 한 후에 input PDF의 histogram과 output PDF인 "hist_matching_PDF"의 histogram을 비교해보면 output PDF histogram의 모양이 reference histogram의 모양과 유사한 것을 파악할 수 있다.

hist_matching_Color.cpp

- 코드 설명

```
8    Mat input = imread("input.jpg", CV_LOAD_IMAGE_COLOR);
9    Mat ref = imread("ref.jpg", CV_LOAD_IMAGE_COLOR);
10   Mat equalized_YUV;
11   Mat equalized_ref;
12
13   cvtColor(input, equalized_YUV, CV_RGB2YUV); // RGB -> YUV
14   cvtColor(ref, equalized_ref, CV_RGB2YUV);   // RGB -> YUV
15
16   // split each channel(Y, U, V)
17   Mat channels[3];
18   split(equalized_YUV, channels);
19   Mat Y = channels[0];                        // U = channels[1], V = channels[2]
20
21   // split each channel(Y, U, V)
22   Mat channels_ref[3];
23   split(equalized_ref, channels_ref);
24   Mat Y_ref = channels_ref[0];                     // U = channels_ref[1], V = channels_ref[2]
```

Color input을 histogram matching하는 코드이므로 RGB Input과 reference를 읽어온다. Color distortion이 발생하지 않도록 하기 위해서는 histogram matching을 Y channel에만 적용해야 하므로 Input과 reference를 읽어서 YUV로 변환 후, Y channel만 분리한다.

```
26   // PDF or transfer function txt files1
27   FILE* f_hist_matching_PDF_YUV, * f_PDF_RGB, *f_PDF_ref_RGB;
28   FILE* f_trans_func_matching_YUV;
29
30   fopen_s(&f_PDF_RGB, "PDF_RGB.txt", "w+");
31   fopen_s(&f_PDF_ref_RGB, "PDF_ref_RGB.txt", "w+");
32   fopen_s(&f_hist_matching_PDF_YUV, "matched_PDF_YUV.txt", "w+");
33   fopen_s(&f_trans_func_matching_YUV, "trans_func_matching_YUV.txt", "w+");
```

Input의 PDF값, reference의 PDF값, histogram matching한 결과의 PDF값, 이때의 transfer function 값을 txt 파일에 저장하기 위하여 변수를 선언하고 파일을 만든다.

```
38   hist_eq(Y, Y, trans_func_eq_YUV, CDF_YUV);      // histogram equalization on Y channel
```

"Y" : RGB input에 대한 equalization이므로 Y channel만을 input으로 사용.

"Y" : histogram equalization된 "Y"를 다시 받아옴.

"trans_func_eq_YUV" : histogram equalization의 transfer function이 들어갈 빈 G형 배열. 0으로 초기화.

"CDF_YUV" : RGB 이미지에 대한 histogram equalization이지만 Y channel 하나만 계산하므로 cal_CDF함수로 CDF 값을 계산한다. 그 결과 값이 들어있는 변수이다. Histogram equalization을 계산할 때 사용된다.

```
43   hist_eq(Y_ref, Y_ref, trans_func_eq_ref, CDF_ref);  // histogram equalization on Y_ref channel
```

"Y_ref" : RGB input에 대한 equalization이므로 Y channel만을 input으로 사용.

"Y_ref" : histogram equalization된 "Y_ref"를 다시 받아옴.

"trans_func_eq_ref" : histogram equalization의 transfer function이 들어갈 빈 G형 배열. 0으로 초기화.

"CDF_ref" : RGB 이미지에 대한 histogram equalization이지만 Y channel 하나만 계산하므로

cal_CDF함수로 CDF 값을 계산한다. 그 결과 값이 들어있는 변수이다. Histogram equalization을 계산할 때 사용된다.

```
47      hist_matching(Y, Y, trans_func_eq_YUV, trans_func_eq_ref, trans_func_matching);
```

"Y" : histogram matching을 할 input을 넣는다.
"Y" : histogram matching된 결과를 다시 Y에 받아온다.
"trans_func_eq_YUV" : "input"를 histogram equalization 할 때의 transfer function.
"trans_func_eq_ref" : "ref"를 histogram equalization할 때의 transfer function.
"trans_func_matching" : histogram matching의 transfer function이 들어갈 빈 G형 배열. 0으로 초기화.

```
94    // histogram equalization
95    void hist_eq(Mat& input, Mat& equalized, G* trans_func, float* CDF) {
96
97        // compute transfer function
98        for (int i = 0; i < L; i++)
99            trans_func[i] = (G)((L - 1) * CDF[i]);
100
101       // perform the transfer function
102       for (int i = 0; i < input.rows; i++)
103           for (int j = 0; j < input.cols; j++)
104               equalized.at<G>(i, j) = trans_func[input.at<G>(i, j)];
105   }
```

Histogram equalization을 계산하기 위한 식은 output=(L-1)*CDF이다. 이 식을 "trans_func"에 넣어준다. "trans_func"은 intensity를 index로 하고 있기 때문에 input의 각 픽셀이 가지는 intensity 값으로 접근할 수 있고, 위에 코드에서 만든 "trans_func"을 이용하여 input의 픽셀들을 histogram equalization해서 "equalized"변수에 넣어준다.

```
107   // histogram matching
108   void hist_matching(Mat& input, Mat& matched, G* trans_func_eq, G* trans_func_ref, G* trans_func_matching) {
109       G temp[L] = { 0 };
110
111       //compute transfer function
112       for (int i = 0; i < L; i++) {
113           for (int j = 0; j < L; j++) {
114               if (i == trans_func_ref[j]) {
115                   temp[i] = j;
116                   break;
```

Trans_func_ref의 역함수를 구하는 코드이다. Y=f(x) 함수를 생각했을 때 trans_func_ref의 index가 x이고, 그 위치에 있는 값은 y이다. 따라서 여기서 역함수를 만들기 위해서는 x와 y의 위치를 바꿔서 x=f(y)로 만들어야 하므로 temp를 이용하여 바꿔준다.

```
118                   if (j == L - 1)
119                       temp[i] = temp[i - 1] + 1;
```

하지만 역함수 x=f(y)에서 x값을 갖지 못하는 y가 발생하면 함수가 아니므로 이를 방지해야 한다. 따라서 만약 그러한 y가 있는 경우 이전 y가 가진 x값에 +1을 한 값을 주도록 했다.

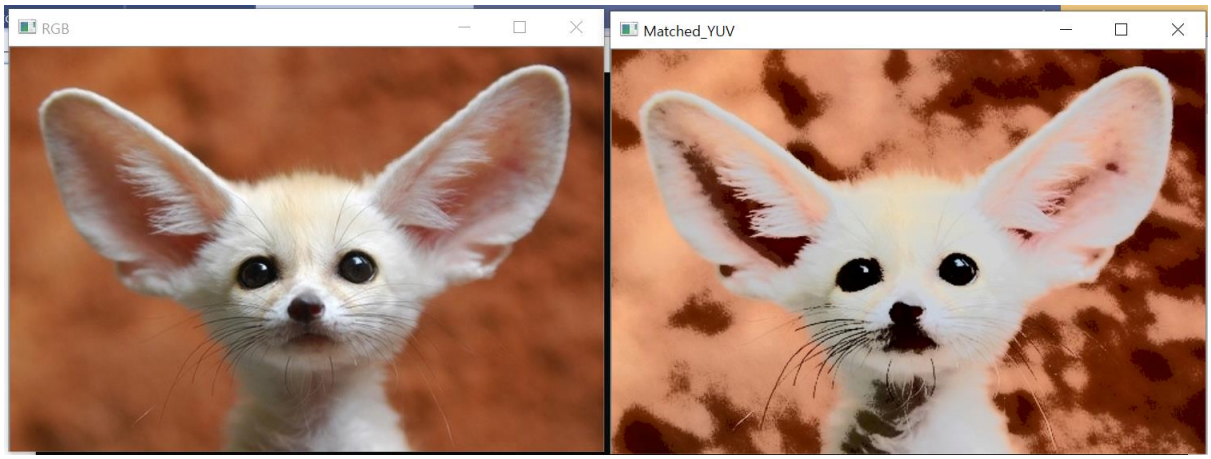
```
123       for (int i = 0; i < L; i++)
124           trans_func_matching[i] = temp[trans_func_eq[i]];
```

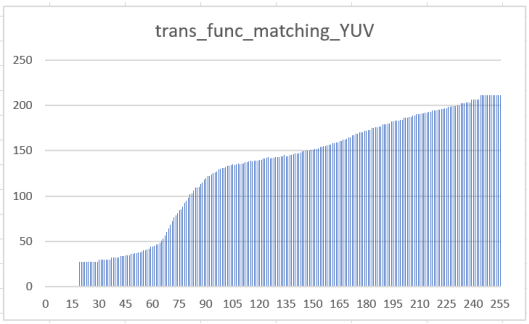
위에서 구한 역함수 temp와 trans_func_eq를 합성하여서 histogram matching을 위한 transfer function, trans_func_matching을 만든다.

```
126         //perform the transfer function
127         for (int i = 0; i < input.rows; i++)
128             for (int j = 0; j < input.cols; j++)
129                 matched.at<G>(i, j) = trans_func_matching[input.at<G>(i, j)];
```

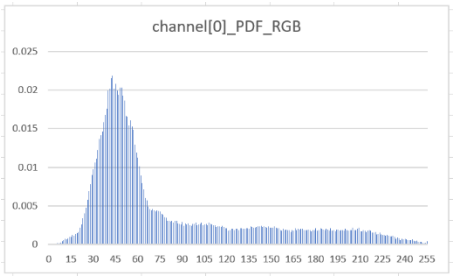
."trans_func_matching"은 intensity를 index로 하고 있기 때문에 input의 각 픽셀이 가지는 intensity 값으로 접근할 수 있고, 위에 코드에서 만든 "trans_func_matching"을 이용하여 input의 픽셀들을 histogram matching해서 "matched"변수에 넣어준다.

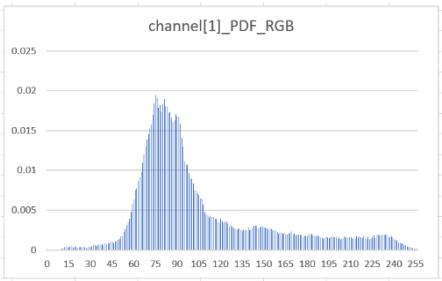
- 결과

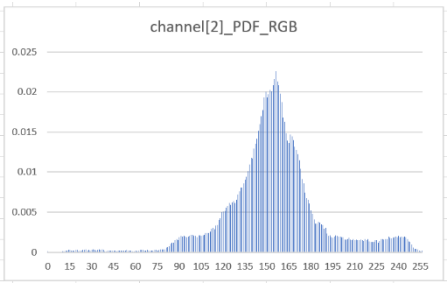




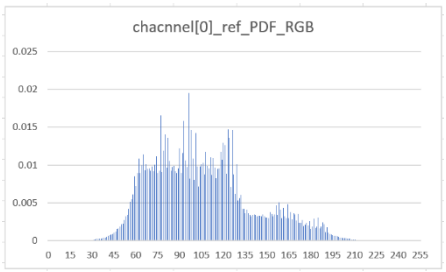
Transfer function for Y channel

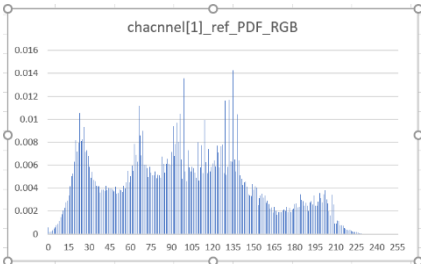


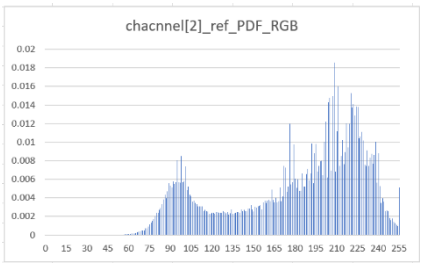




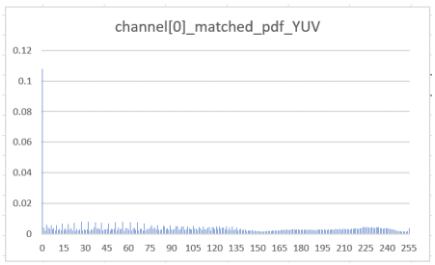
histograms of the original image for RGB

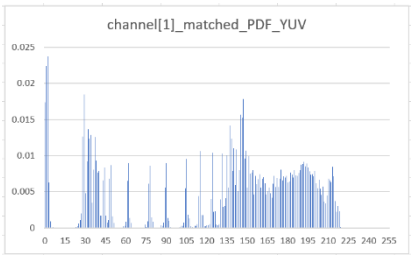


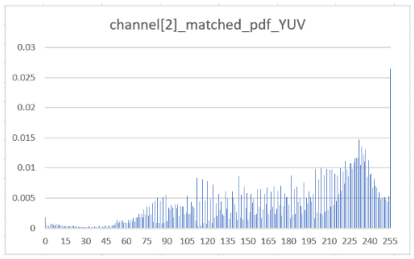




histograms of the reference image for RGB







histograms of the output image for RGB

- 분석

Color input으로 histogram matching을 하는 경우 histogram equalization을 할 때와 마찬가지로 RGB를 YUV로 변환하여서 Y channel에만 transfer해야 한다. histogram matching을 한 후에 input PDF의 histogram과 output PDF인 "hist_matching_PDF_YUV"의 histogram을 비교해보면 output PDF histogram의 모양이 reference histogram의 모양과 유사한 것을 파악할 수 있다.