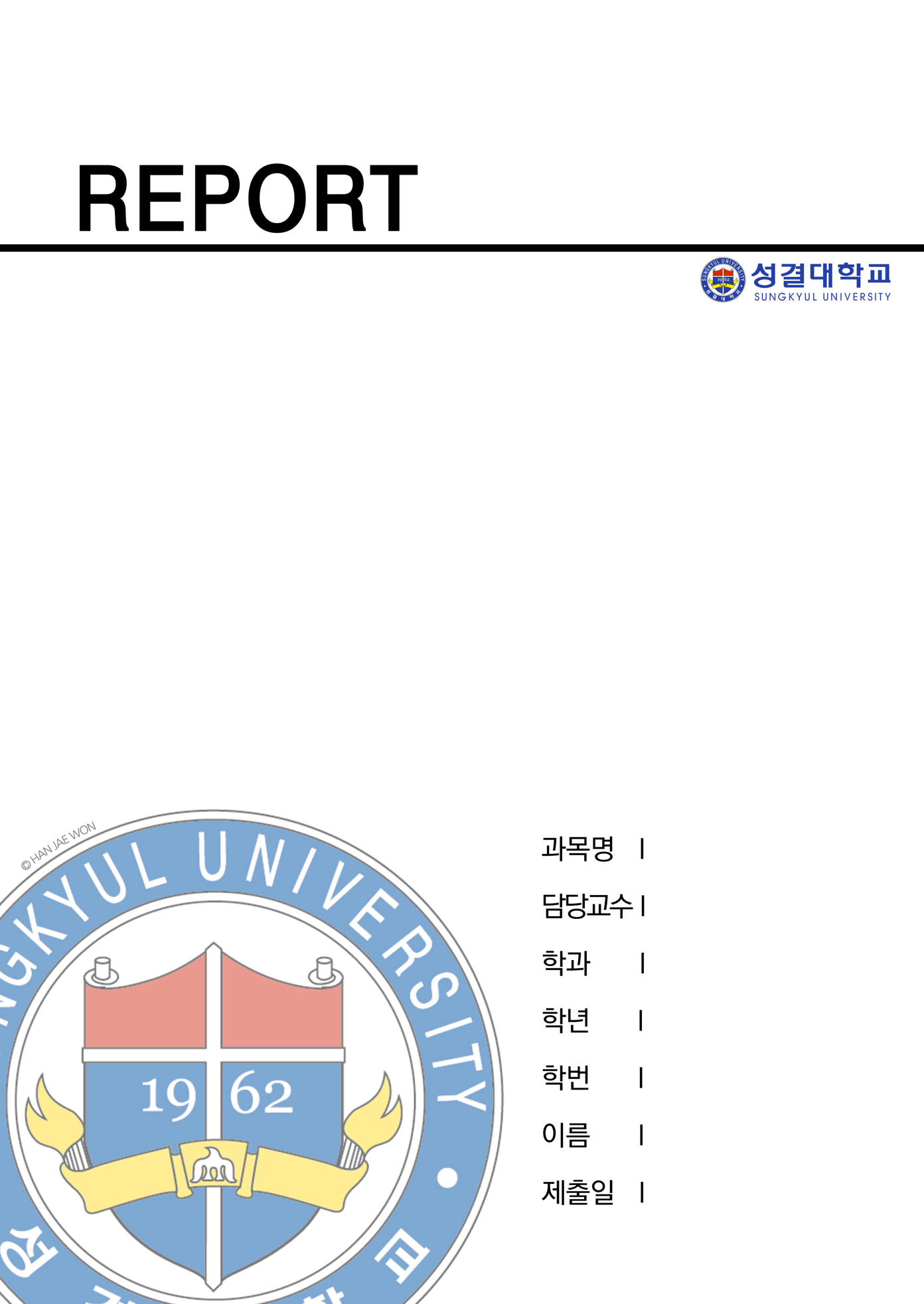


영상처리

1. Histogram Equalization

2. Histogram Specification

20131000 김찬영 영상처리 오후반



1. Histogram Equalization

## 1)코딩부분

//get\_hist에서 원본 히스토그램,CDF 를 구한뒤 평활화 작업을 해준다.  
//get\_Equalhist에서 평활화 후 히스토그램,CDF를 구한다  
  
//원본 히스토그램,CDF를 구한 후 평활화작업  
**void** get\_hist(uchar\*\* img, uchar\*\* result\_img, **int** X\_Size, **int** Y\_Size)  
{  
 **int** i, j, tmp;  
 **double** compare;  
 **int** t, tp, range, hrange;  
 CvSize histoSize, cdfSize;  
 IplImage\* imgHisto, \* cdfImgHisto;  
 //히스토그램 사이즈  
 histoSize.width = 256;  
 histoSize.height = 256;  
 // CDF 사이즈  
 cdfSize.width = 256;  
 cdfSize.height = 256;  
 //원본 히스토그램  
 imgHisto = cvCreateImage(histoSize, 8, 1);  
 //원본 CDF  
 cdfImgHisto = cvCreateImage(cdfSize, 8, 1);  
 //히스토그램 초기화  
 **for** (i = 0; i < histoSize.height; i++)  
 **for** (j = 0; j < histoSize.width; j++)  
 {  
 ((uchar\*)(imgHisto->imageData + imgHisto->widthStep \* i))[j] = 0;  
 }  
 //CDF 초기화  
 **for** (i = 0; i < cdfSize.height; i++)  
 **for** (j = 0; j < cdfSize.width; j++)  
 {  
 ((uchar\*)(cdfImgHisto->imageData + cdfImgHisto->widthStep \* i))[j] = 0;  
 }  
  
 tp = X\_Size \* Y\_Size;  
 //히스토그램 변수 모두 초기화  
 **for** (i = 0; i < 256; i++)  
 histogram[i] = 0;  
  
 **for** (i = 0; i < Y\_Size; i++)  
 **for** (j = 0; j < X\_Size; j++)  
 histogram[img[i][j]]++;  
  
 compare = 0;  
 //히스토그램 양수,음수 비교  
 **for** (i = 0; i < 256; ++i)  
 {  
 compare = compare > histogram[i] ? compare : histogram[i];  
 }  
 // 원본 히스토그램  
 **for** (i = 0; i < 256; ++i)  
 {  
 tmp = (**int**)255 \* (histogram[i] / compare);  
 cvLine(imgHisto, cvPoint(i, 255), cvPoint(i, 255 - tmp), CV\_RGB(255, 255, 255), 1, 8, 0);  
 }  
 cvShowImage("Original Histogram", imgHisto);  
 //CDF 시작  
 cdfOfHisto[0] = histogram[0];  
  
 **for** (i = 0; i < 256; i++)  
 cdfOfHisto[i] = cdfOfHisto[i - 1] + histogram[i];  
  
  
 compare = (**double**)cdfOfHisto[255];  
 **for** (i = 0; i < 256; ++i)  
 {  
 tmp = (**int**)255 \* (cdfOfHisto[i] / compare);  
  
 cvLine(cdfImgHisto, cvPoint(i, 255), cvPoint(i, 256 - tmp), CV\_RGB(255, 255, 255), 1, 8, 0);  
 }  
 cvShowImage("CDF of Original Image", cdfImgHisto);  
 //역함수를 이용해 평활화 작업  
 range = cdfOfHisto[255] - cdfOfHisto[0];  
 histogramEqual[0] = 0;  
 **for** (i = 1; i < 256; ++i) {  
 t = (**int**)ceil(((cdfOfHisto[i] - cdfOfHisto[0]) \* 255.0) / range);  
 histogramEqual[i] = (t < 0) ? 0 : (t > 255) ? 255 : t;  
 }  
 cvReleaseImage(&imgHisto);  
 **for** (i = 0; i < Y\_Size; ++i)  
 **for** (j = 0; j < X\_Size; ++j)  
 result\_img[i][j] = histogramEqual[img[i][j]];  
}

}  
}

// 평활화 후 히스토그램과 CDF  
**void** get\_Equalhist(uchar\*\* img, **int** X\_Size, **int** Y\_Size)  
{  
 **int** i, j, tmp;  
 **double** compare;  
 **int** t, tp, range, hrange;  
 CvSize histoSize, cdfSize;  
 IplImage\* imgHisto, \* cdfImgHisto;  
 //히스토그램 사이즈  
 histoSize.width = 256;  
 histoSize.height = 256;  
 //CDF 사이즈  
 cdfSize.width = 256;  
 cdfSize.height = 256;  
 //평활화 후 히스토그램  
 imgHisto = cvCreateImage(histoSize, 8, 1);  
 //평활화 후 CDF  
 cdfImgHisto = cvCreateImage(cdfSize, 8, 1);  
 //히스토그램 초기화  
 **for** (i = 0; i < histoSize.height; i++)  
 **for** (j = 0; j < histoSize.width; j++)  
 {  
 ((uchar\*)(imgHisto->imageData + imgHisto->widthStep \* i))[j] = 0;  
 }  
 //CDF 초기화  
 **for** (i = 0; i < cdfSize.height; i++)  
 **for** (j = 0; j < histoSize.width; j++)  
 {  
 ((uchar\*)(cdfImgHisto->imageData + cdfImgHisto->widthStep \* i))[j] = 0;  
 }  
  
 tp = X\_Size \* Y\_Size;  
 //히스토그램 변수 모두 초기화  
 **for** (i = 0; i < 256; i++)  
 histogram[i] = 0;  
  
 **for** (i = 0; i < Y\_Size; i++)  
 **for** (j = 0; j < X\_Size; j++)  
 histogram[img[i][j]]++;  
  
  
 compare = 0;  
 //히스토그램 양수,음수 비교  
 **for** (i = 0; i < 256; ++i)  
 {  
 compare = compare > histogram[i] ? compare : histogram[i];  
 }  
 // 평활화 후 히스토그램  
 **for** (i = 0; i < 256; ++i)  
 {  
 tmp = (**int**)255 \* (histogram[i] / compare);  
  
 cvLine(imgHisto, cvPoint(i, 255), cvPoint(i, 255 - tmp), CV\_RGB(255, 255, 255), 1, 8, 0);  
 }  
  
 cvShowImage("Histogram after Equalization", imgHisto);  
  
 // 평활화 후 CDF  
 cdfOfHisto[0] = histogram[0];  
 **for** (i = 1; i < 256; i++)  
 {  
 cdfOfHisto[i] = cdfOfHisto[i - 1] + histogram[i];  
 }  
  
 compare = (**double**)cdfOfHisto[255];  
 **for** (i = 0; i < 256; ++i)  
 {  
 tmp = (**int**)255 \* (cdfOfHisto[i] / compare);  
  
 cvLine(cdfImgHisto, cvPoint(i, 255), cvPoint(i, 256 - tmp), CV\_RGB(255, 255, 255), 1, 8, 0);  
 }  
  
 cvShowImage("CDF after Equalization", cdfImgHisto);  
  
 //역함수를 이용해 평활화 작업  
 range = cdfOfHisto[255] - cdfOfHisto[0];  
 histogramEqual[0] = 0;  
 **for** (i = 0; i < 256; ++i) {  
 t = (**int**)ceil(((cdfOfHisto[i] - cdfOfHisto[0]) \* 255.0) / range);  
 histogramEqual[i] = (t < 0) ? 0 : (t > 255) ? 255 : t;  
 }  
  
 cvReleaseImage(&imgHisto);  
 cvReleaseImage(&cdfImgHisto);  
 **for** (i = 0; i < Y\_Size; ++i)  
 **for** (j = 0; j < X\_Size; ++j)  
 img[i][j] = histogramEqual[img[i][j]];  
}

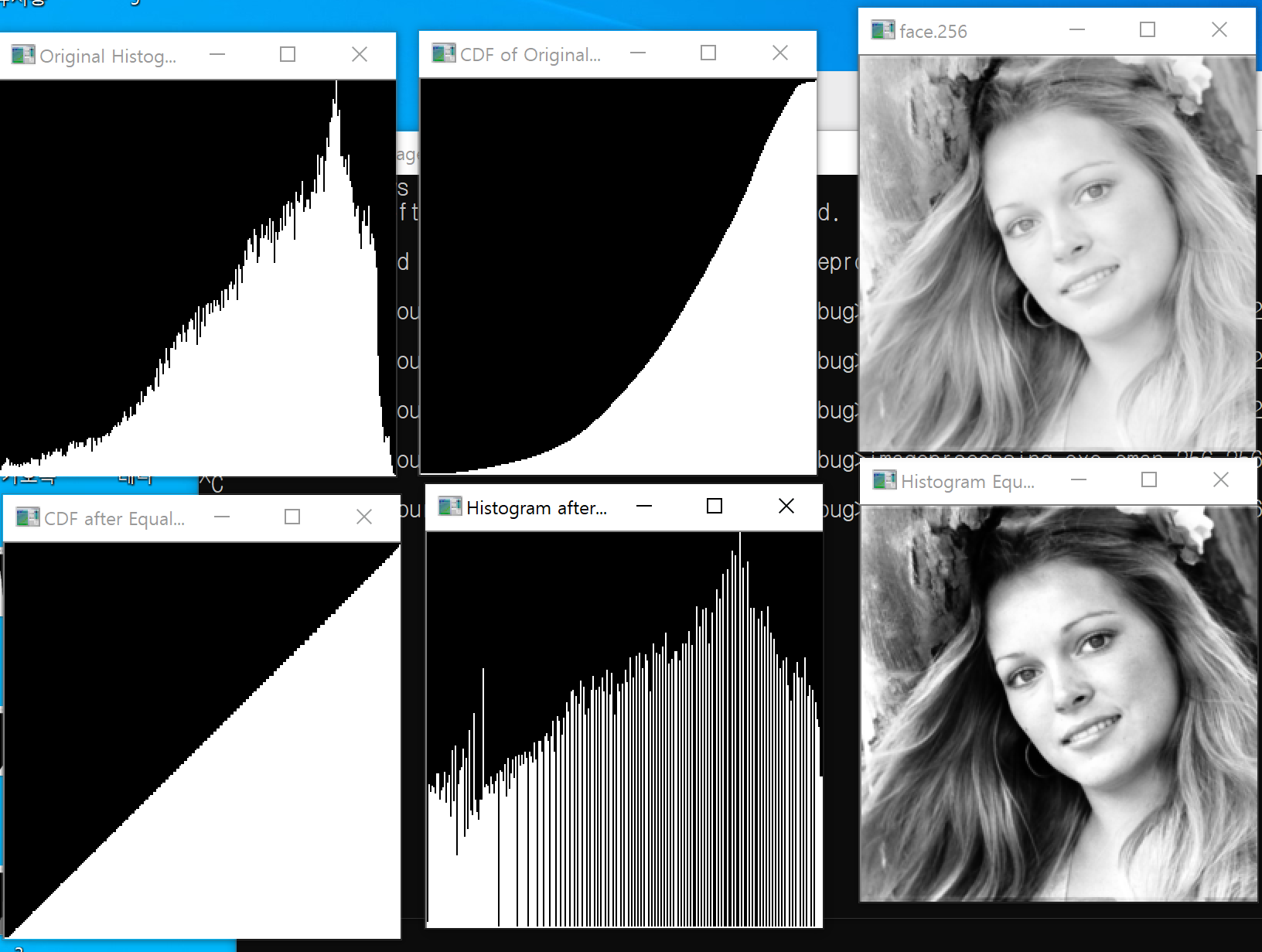
## 2) 실행화면

ㄱ.코드실행시cmd에서 imageprocessing.exe lena.512 512 512 했을시

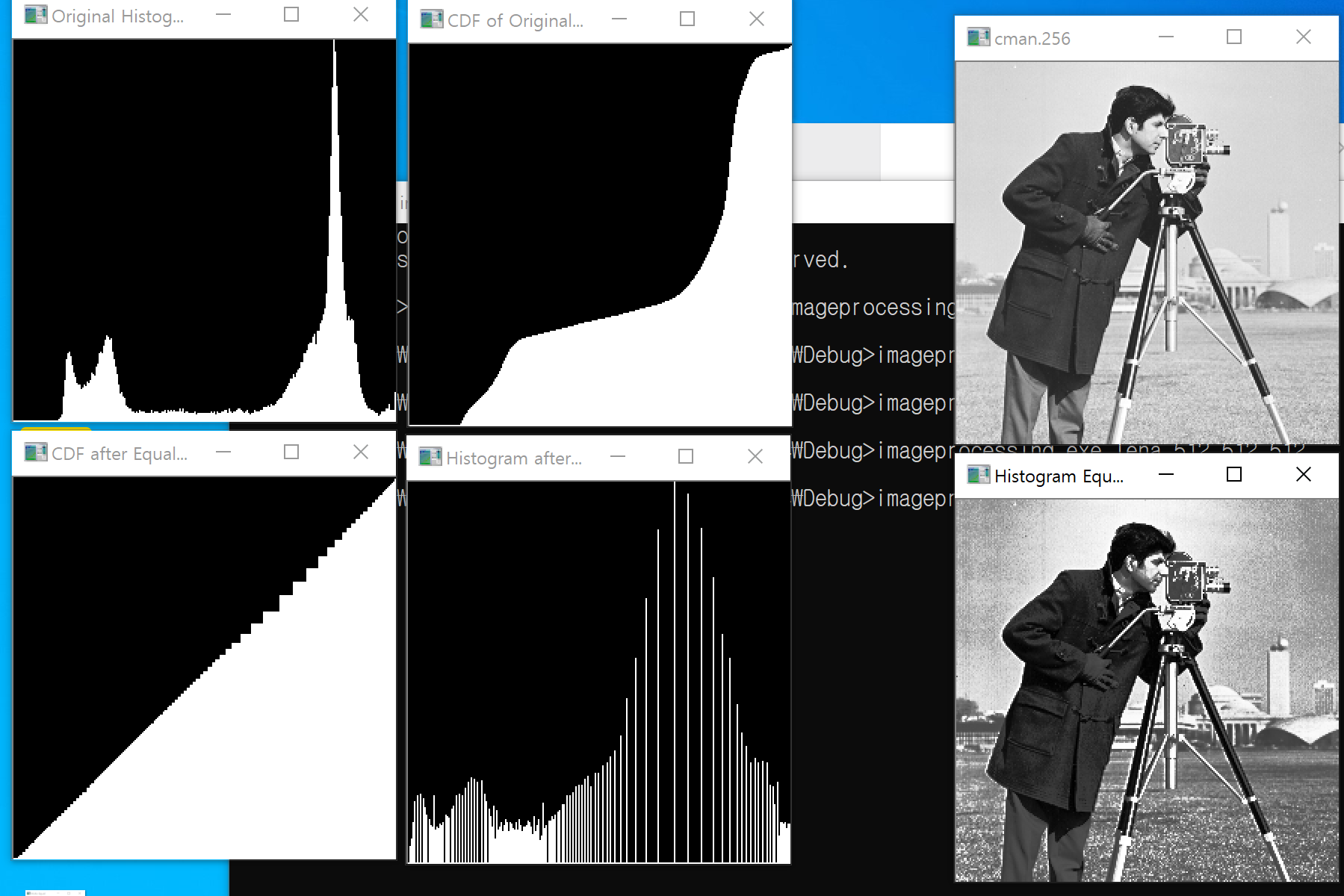
사람, 여자, 건물, 사진이(가) 표시된 사진

자동 생성된 설명

ㄴ.코드실행시cmd에서 imageprocessing.exe face.256 256 256 했을시



ㄷ.코드실행시cmd에서 imageprocessing.exe cman.256 256 256 했을시



2. Histogram Specification

## 1)코딩부분

**//원본을 비교할이미지 감마에 맞춤.**

**void** get\_Match(uchar\*\* img, uchar\*\* result\_img, **int** x\_size, **int** y\_size, **int** histogram[256], **int** histogramSpec[256], **int** histogramMatch[256])  
{  
 **int** i, j, tmp, matchz = 0;  
 **float** diff;  
 printf("Start HistoGram Specification \n");  
 **for** (i = 0; i < 256; i++)  
 {  
 //초기화  
 histogramMatch[i] = 0;  
 //비교누적분포함수  
 **for** (j = 0; j < 256; j++)  
 {  
 **if** ((i - histogramSpec[j]) > 0)  
 {  
 histogramMatch[i] = j;  
 }  
 }  
 }  
 **for** (i = 0; i < y\_size; ++i)  
 **for** (j = 0; j < x\_size; ++j)  
 result\_img[i][j] = histogramMatch[img[i][j]];  
}

## 2)코드실행시

cmd에서 imegeprocessing.exe face.256 256 256 cman.256 256 256 했을시



*비교할 이미지가 비교적 밝기 때문에 원본의 CDF가 밝아진다.*