CODE:

#include <iostream>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/imgproc/imgproc.hpp>

using std::cout;

using std::cin;

using std::endl;

using namespace cv;

void imhist(Mat image, int histogram[])

{

for(int i = 0; i < 256; i++)

{

histogram[i] = 0;

}

for(int y = 0; y < image.rows; y++)

for(int x = 0; x < image.cols; x++)

histogram[(int)image.at<uchar>(y,x)]++;

}

void cumhist(int histogram[], int cumhistogram[])

{

cumhistogram[0] = histogram[0];

for(int i = 1; i < 256; i++)

{

cumhistogram[i] = histogram[i] + cumhistogram[i-1];

}

}

void histDisplay(int histogram[], const char\* name)

{

int hist[256];

for(int i = 0; i < 256; i++)

{

hist[i]=histogram[i];

}

int hist\_w = 512; int hist\_h = 400;

int bin\_w = cvRound((double) hist\_w/256);

Mat histImage(hist\_h, hist\_w, CV\_8UC1, Scalar(255, 255, 255));

int max = hist[0];

for(int i = 1; i < 256; i++){

if(max < hist[i]){

max = hist[i];

}

}

for(int i = 0; i < 256; i++){

hist[i] = ((double)hist[i]/max)\*histImage.rows;

}

for(int i = 0; i < 256; i++)

{

line(histImage, Point(bin\_w\*(i), hist\_h),

Point(bin\_w\*(i), hist\_h - hist[i]),

Scalar(0,0,0), 1, 8, 0);

}

namedWindow(name, CV\_WINDOW\_AUTOSIZE);

imshow(name, histImage);

}

int main()

{

Mat image = imread("scene.jpg", CV\_LOAD\_IMAGE\_GRAYSCALE);

int histogram[256];

imhist(image, histogram);

int size = image.rows \* image.cols;

float alpha = 255.0/size;

float PrRk[256];

for(int i = 0; i < 256; i++)

{

PrRk[i] = (double)histogram[i] / size;

}

int cumhistogram[256];

cumhist(histogram,cumhistogram );

int Sk[256];

for(int i = 0; i < 256; i++)

{

Sk[i] = cvRound((double)cumhistogram[i] \* alpha);

}

float PsSk[256];

for(int i = 0; i < 256; i++)

{

PsSk[i] = 0;

}

for(int i = 0; i < 256; i++)

{

PsSk[Sk[i]] += PrRk[i];

}

int final[256];

for(int i = 0; i < 256; i++)

final[i] = cvRound(PsSk[i]\*255);

Mat new\_image = image.clone();

for(int y = 0; y < image.rows; y++)

for(int x = 0; x < image.cols; x++)

new\_image.at<uchar>(y,x) = saturate\_cast<uchar>(Sk[image.at<uchar>(y,x)]);

namedWindow("Original Image");

imshow("Original Image", image);

histDisplay(histogram, "Original Histogram");

namedWindow("Linear filtered Image");

imshow("Linear filtered ",new\_image);

histDisplay(final, " Linear filtered Histogram");

waitKey();

return 0;

}