**First Round - Final Code Design Document (PFA)**

GIVEN PROBLEM:

Implement a Histogram equalization from scratch using C++.  Input should be an Image and the output should be a Linear Filtered Image, Neat Documentation is expected with Code, Explanation, Input, and Output Image.

HISTOGRAM EQUALIZATION:

Histogram equalization is a method in [image processing](https://en.wikipedia.org/wiki/Image_processing) of [contrast](https://en.wikipedia.org/wiki/Contrast_(vision)) adjustment using the [image](https://en.wikipedia.org/wiki/Image)'s [histogram](https://en.wikipedia.org/wiki/Image_histogram). Histogram equalization is the best method for image enhancement. It provides better quality of images without loss of any information.

HOW DOES IT WORK?

Equalization implies mapping one distribution (the given histogram) to another distribution (a wider and more uniform distribution of intensity values) so the intensity values are spread over the whole range.

To accomplish the equalization effect, the remapping should be the **cumulative distribution function (cdf)**. For the histogram H(i), its **cumulative distribution H′(i)** is:

H′(i)=∑0≤j<iH(j)

To use this as a remapping function, we have to normalize H′(i) such that the maximum value is 255

SOLUTION:

The problem is solved using Visual Studio IDE and openCV library which is used to read and analyse image. The openCV package is downloaded from the following site.

<https://sourceforge.net/projects/opencvlibrary/files/4.4.0/opencv-4.4.0-vc14_vc15.exe/download>.

After the installation of openCV, its path is added in the Visual studio and all the dependencies are imported.

SOURCE 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 imgh(Mat image, int histogram[])

{

// initialize all intensity values to 0

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

{

histogram[i] = 0;

}

// calculate the no of pixels for each intensity values

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

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

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

}

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

{

cumhistogram[0] = histogram[0];

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

{

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

}

}

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

{

int hist[256];

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

{

hist[i] = histogram[i];

}

// draw the histograms

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));

// find the maximum intensity element from histogram

int max = hist[0];

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

if (max < hist[i]) {

max = hist[i];

}

}

// normalize the histogram between 0 and histImage.rows

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

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

}

// draw the intensity line for histogram

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);

}

// display histogram

namedWindow(name, WINDOW\_AUTOSIZE);

imshow(name, histImage);

}

int main()

{

// Load the image

Mat image = imread("C:\\Projects\\curneu\_technical\_assessment\\hist\\apj.jpg", IMREAD\_GRAYSCALE);

// Generate the histogram

int histogram[256];

imgh(image, histogram);

// Caluculate the size of image

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

float alpha = 255.0 / size;

// Calculate the probability of each intensity

float PrRk[256];

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

{

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

}

// Generate cumulative frequency histogram

int cumhistogram[256];

ch(histogram, cumhistogram);

// Scale the histogram

int Sk[256];

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

{

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

}

// Generate the equlized histogram

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);

// Generate the equlized image

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)]);

// Display the original Image

namedWindow("Original Image");

imshow("Original Image", image);

// Display the original Histogram

hDisplay(histogram, "Original Histogram");

// Display equilized image

namedWindow("Equilized Image");

imshow("Equilized Image", new\_image);

// Display the equilzed histogram

hDisplay(final, "Equilized Histogram");

waitKey();

return 0;

}

OUTPUT





