OpenCV Tutorial C++

Histogram Equalization of Grayscale or Color Image

Histogram

Histogram is the intensity distribution of an image.

E.G -
Consider the following image. Say, depth of the image is 2 bits. Therefore the value range for each and every pixel is from 0 to 3.

2	3	2	1	2
1	2	0	2	3
0	1	1	3	1
2	3	0	1	2
0	1	2	2	0

Sample Image (Depth = 2 bits)

Histogram of the a image shows how the pixel values are distributed. As you can see in the above image there are 5 pixels with value 0, 7 pixels with value 1, 9 pixels with value 2 and 4 pixels with value 3. These information is tabulated as follows.

Pixel Value	Number of Pixels
0	5
1	7
2	9
3	4

Intensity Distribution of above image

Histogram of a image usually presented as a graph. The following graph represents the histogram of the above image.

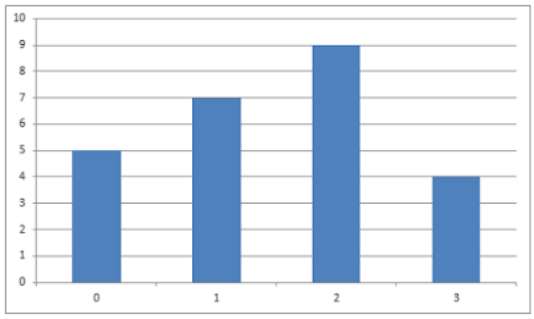
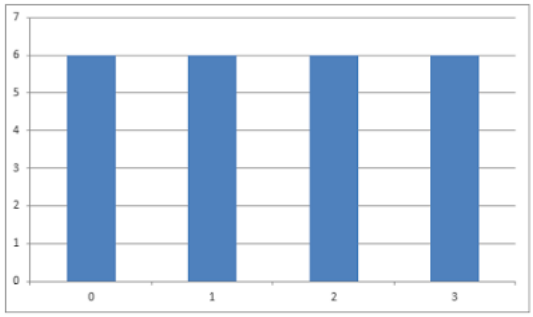


Image Histogram

Histogram Equalization

Histogram Equalization is defined as equalizing the intensity distribution of an image or flattening the intensity distribution curve. Histogram equalization is used to improve the contrast of an image. The equalized histogram of the above image should be ideally like the following graph.



Equalized Histogram

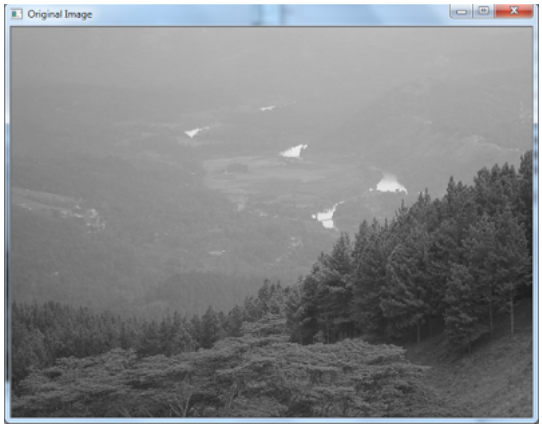
But practically, you cannot achieve this kind of perfect histogram equalization. But there are various techniques to achieve histogram equalization close to the perfect one. In OpenCV, there is an in-built OpenCV function to equalize histogram.

Histogram Equalization of Grayscale image

Here is the sample program demonstrating how to equalize the histogram of a grayscale image (black and white image) using a OpenCV in-built function.

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, const char** argv )
{
    Mat img = imread("MyPic.JPG", CV_LOAD_IMAGE_COLOR); //open and read the image

    if (img.empty())
    {
        cout << "Image cannot be loaded..!!" << endl;
        return -1;
    }

    cvtColor(img, img, CV_BGR2GRAY); //change the color image to grayscale image

    Mat img_hist_equalized;
    equalizeHist(img, img_hist_equalized); //equalize the histogram

    //create windows
    namedWindow("Original Image", CV_WINDOW_AUTOSIZE);
    namedWindow("Histogram Equalized", CV_WINDOW_AUTOSIZE);

    //show the image
    imshow("Original Image", img);
    imshow("Histogram Equalized", img_hist_equalized);

    waitKey(0); //wait for key press

    destroyAllWindows(); //destroy all open windows

    return 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

You can download this OpenCV visual c++ project from here. (The downloaded file is a compressed .rar folder. So, you have to extract it using Winrar or other suitable software)



Original Image

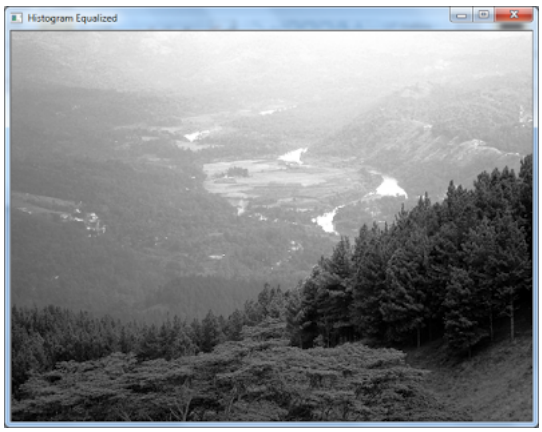


Image with Equalized Histogram

New OpenCV functions

- **void cvtColor(InputArray src, OutputArray dst, int code, int dstCn=0)**

This function converts image from one color space to another color space.
OpenCV usually loads an image in BGR color space. In the above example, I want to change the image to grayscale color space. So, I use the **CV_BGR2GRAY** as the 3rd parameter. If you want to convert to HSV color space, you should use **CV_BGR2HSV.**

This is an explanation of each parameters of the above function.

- **InputArray src**- Input image (it should be 8 bit unsigned or 16 bit unsigned or 32 bit floating point image)
- **OutputArray dst** - Output image (It should have a same size and depth as the source image)
- **int code**- Should specify the color space conversion. There are many codes available. Here are some of them.
  - CV_BGR2HSV
  - CV_HSV2BGR
  - CV_RGB2HLS
  - CV_HLS2RGB
  - CV_BGR2GRAY
  - CV_GRAY2BGR
- **int dstCn -** Number of channels in the destination image. If it is 0, number of channels of the destination image is automatically derived from source image and color space conversion code. For a beginner, it is recommended to use 0 for this parameter.

- **void equalizeHist(InputArray src, OutputArray dst)**

This function equalizes the histogram of a single channel image (Grayscale image is a single channel image)
By equalizing the histogram, the brightness is normalized. As a result, the contrast is improved.

Here is the description of each parameters of the above OpenCV function.

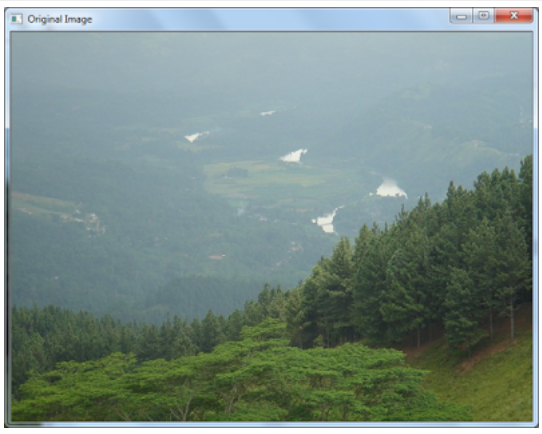
- **InputArray src** - 8 bit single channel image
- **OutputArray dst -** Destination image of which histogram is equalized (It should have the same size and depth as the source image.)

Histogram Equalization of Color image

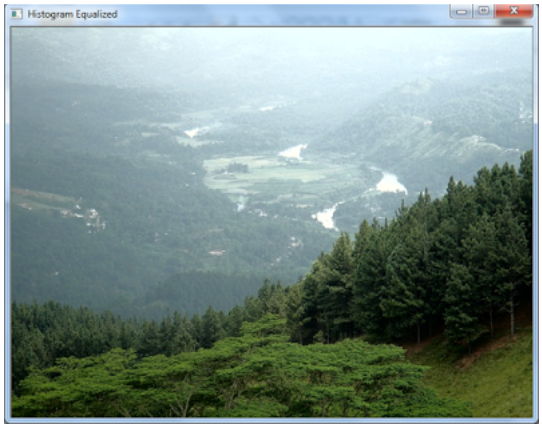
In the above example, I have shown how to equalize the histogram of a grayscale image. Now I am going to show you how to equalize histogram of a color image using sample OpenCV program.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main( int argc, const char** argv )
{
    Mat img = imread("MyPic.JPG", CV_LOAD_IMAGE_COLOR); //open and read the image

    if (img.empty()) //if unsuccessful, exit the program
    {
        cout << "Image cannot be loaded..!!" << endl;
        return -1;
    }

    vector<Mat> channels;
    Mat img_hist_equalized;

    cvtColor(img, img_hist_equalized, CV_BGR2YCrCb); //change the color image from BGR to YCrCb format

    split(img_hist_equalized,channels); //split the image into channels

    equalizeHist(channels[0], channels[0]); //equalize histogram on the 1st channel (Y)

  merge(channels,img_hist_equalized); //merge 3 channels including the modified 1st channel into one image

     cvtColor(img_hist_equalized, img_hist_equalized, CV_YCrCb2BGR); //change the color image from YCrCb to BGR format (to display image properly)

    //create windows
    namedWindow("Original Image", CV_WINDOW_AUTOSIZE);
    namedWindow("Histogram Equalized", CV_WINDOW_AUTOSIZE);

    //show the image
    imshow("Original Image", img);
    imshow("Histogram Equalized", img_hist_equalized);

    waitKey(0); //wait for key press

    destroyAllWindows(); //destroy all open windows

    return 0;
}
////////////////////////////////////////////////////////////////////////////////////////////////////
```

You can download this OpenCV visual c++ project from here. (The downloaded file is a compressed .rar folder. So, you have to extract it using Winrar or other suitable software)



Original Color Image



Color Image with Equalized Histogram

New OpenCV functions

- **cvtColor(img, img_hist_equalized, CV_BGR2YCrCb)**

This line converts the color space of BGR in '**img**' to YCrCb color space and stores the resulting image in '**img_hist_equalized**'.

In the above example, I am going to equalize the histogram of color images. In this scenario, I have to equalize the histogram of the intensity component only, not the color components. So, BGR format cannot be used because its all three planes represent color components blue, green and red. So, I have to convert the original BGR color space to YCrCb color space because its 1st plane represents the intensity of the image where as other planes represent the color components.

- **void split(const Mat& m, vector<Mat>& mv)**

This function splits each channel of the '**m**' multi-channel array into separate channels and stores them in a vector, referenced by '**mv**'.

Argument list

  - **const Mat& m -** Input multi-channel array
  - **vector<Mat>& mv -** vector that stores the each channel of the input array

- **equalizeHist(channels[0], channels[0]);**

Here we are only interested in the 1st channel (Y) because it represents the intensity information whereas other two channels (Cr and Cb) represent color components. So, we equalize the histogram of the 1st channel using OpenCV in-built function, '**equalizeHist**(..)' and other two channels remain unchanged.

- **void merge(const vector<Mat>& mv, OutputArray dst)**

This function does the reverse operation of the split function. It takes the vector of channels and create a single multi-channel array.
Argument list

  - **const vector<Mat>& mv** - vector that holds several channels. All channels should have same size and same depths
  - **OutputArray dst** - stores the destination multi-channel array

- **cvtColor(img_hist_equalized, img_hist_equalized, CV_YCrCb2BGR)**

This line converts the image from YCrCb color space to BGR color space. It is essential to convert to BGR color space because '**imshow**(..)' OpenCV function can only show images with that color space.

This is the end of the explanation of new OpenCV functions, found in the above sample code. If you are not familiar with other OpenCV functions, please refer to the previous lessons.

Next Lesson: Smooth / Blur Images
Previous Lesson: Change Contrast of Image or Video

7 comments: