Multimedia
(Lab 03)

Spring, 2020

Department of Software
Yong Ju Jung (정용주)

가천대학교 소프트웨어학과
Gachon University

Summary

- In Previous Lab, you have exercised a basic pixel processing of images.

- In this lab, you will learn simple image processing techniques.
  - histogram equalization
  - image filtering techniques.

[Lab03-1]

- Histogram equalization to enhance the contrast of an image
  - Load a color image (using **cv::imread**)
  - Convert the original image to grayscale
    - cvtColor(src, src, CV_BGR2GRAY);
  - Equalize the Histogram by using the OpenCV function EqualizeHist
    - equalizeHist(src, dst);
  - Display the source and equalized images in a window (using **cv::imshow**)

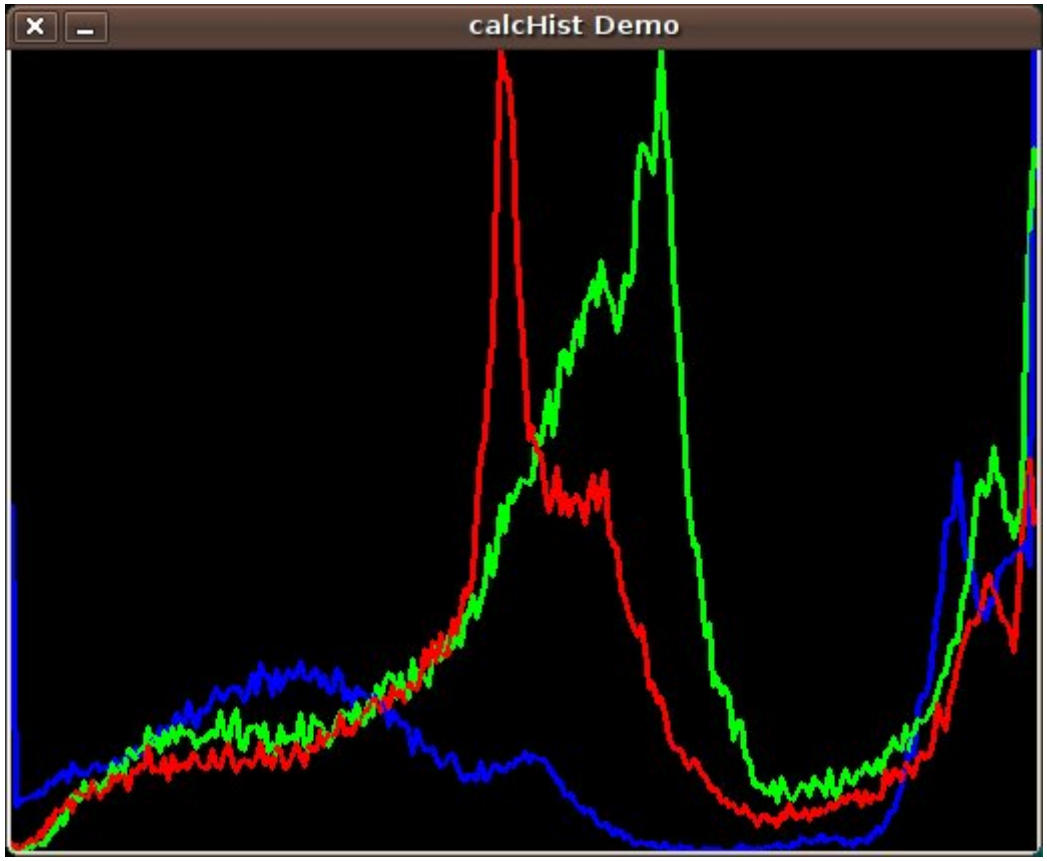
```
//Lab3 - 1: histogram equalization
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <stdio.h>

using namespace cv;
using namespace std;

/** @function main */
int main( int argc, char** argv )
{
  Mat src, dst;

  char* source_window = "Source image";
  char* equalized_window = "Equalized Image";

  /// Load image
  src = imread( argv[1], 1 );

  if( !src.data )
    { cout<<"Usage: ./Histogram_Demo <path_to_image>"<<endl;
      return -1;}

  /// Convert to graysoale
  cvtColor( src, src, CV_BGR2GRAY );

  /// Apply Histogram Equalization
  equalizeHist( src, dst );

  /// Display results
  namedWindow( source_window, CV_WINDOW_AUTOSIZE );
  namedWindow( equalized_window, CV_WINDOW_AUTOSIZE );

  imshow( source_window, src );
  imshow( equalized_window, dst );

  /// Wait until user exits the program
  waitKey(0);

  return 0;
}
```

```
string srcImg_Path = "D:₩₩repos_VS₩₩Project_sourceIMG₩₩Lena_color.
string equalizedImg_Path = "D:₩₩repos_VS₩₩Project_sourceIMG₩
₩lena_noise.png";
        //Load image
        if (argc > 1) {
                srcImg_Path = argv[1];
                equalizedImg_Path = argv[2];
        }
        Mat src, dst;
        src = imread(srcImg_Path.c_str(), IMREAD_COLOR);
        if (src.empty()) {
                cout << "Could not open or find the image1" << std::endl;
                return -1;
        }
        //convert to gray scale
        cvtColor(src, src, COLOR_BGR2GRAY);
        //apply histogram equalization
        equalizeHist(src, dst);
        //Display results
        imshow("source Image", src);
        imshow("equalized Image", dst);
        //Wait until user exits program
        waitKey(0);
        return 0;
```

Extension of Lab03-1

- Plot two histograms
  - One for input image
  - The other for output image after histogram equalization

Histogram Calculation & Plot



Histogram Calculation

- Calculate & plot the histogram of your input image.

- calcHist(&image, 1, 0, Mat(), hist, 1, &histSize, &histRange, uniform, accumulate);
  - Mat image; // input image
  - Mat hist; // output histogram
  - int histSize = 256; //from 0 to 255
  - Set the range of values (as we said, between 0 and 255)
    - float range[] = { 0, 256 } ; //the upper boundary is exclusive
    - const float* histRange = { range };
  - bool uniform = true;
  - bool accumulate = false;

Draw a histogram

- // Draw the histogram for a grayscale image
- int hist_w = 512; int hist_h = 400;
- int bin_w = cvRound((double) hist_w/histSize);
- Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(0,0,0));

- normalize(hist, hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());

- for(int i = 1; i < histSize; i++)
  line(histImage, Point(bin_w*(i-1), hist_h -
  cvRound(hist.at<float>(i-1))) , Point(bin_w*(i), hist_h -
  cvRound(hist.at<float>(i))), Scalar(255, 0, 0), 2, 8, 0);

- namedWindow("calcHist Demo", WINDOW_AUTOSIZE);
- imshow("calcHist Demo", histImage);

https://docs.opencv.org/3.1.0/d8/dbc/tutorial_histogram_calculation.html

[Lab03-2]

- Smoothing filter for an image
  - Download an image from E-Class. The file is Lena in gray-scale.
  - Load the image (using **cv::imread**)
  - Perform NxN smoothing (average) filtering for the image, as you learn in this lecture (see Figure in page 40 in lecture 08).
  - Display the result image in an OpenCV window (using **cv::imshow**)

[Lab03-3]

- Median filter for an image
  - Download an image from E-Class. The file is the noisy Lena image.
  - Load the image (using **cv::imread**)
  - Perform median filtering with NxN window for the image, as you learn in this lecture (slide #40 in lecture 08).
  - Display the result image in an OpenCV window (using **cv::imshow**)

3*3 median filtering

* Source from Wikipedia

[Lab03-4]

- Derivative filter to obtain image gradient for an image
  - Download a gray-scale image.
  - Load the image (using **cv::imread**).
  - Perform 3x3 **Sobel** filtering for the image.
  - Display the gradient magnitude map of the filter output in an OpenCV window (using **cv::imshow**)

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

[Lab03-5]

';여기서 F''(x)에 절대값을 씌우면 68페이지에서처럼 극점이 2개가 돼서 필터된 엣지가 두줄이 나옴.

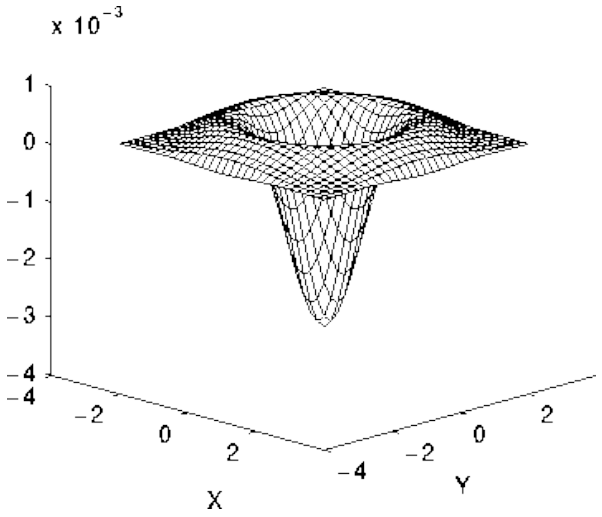
- Sharpening filter by using Laplacian filter
  - Download an image from E-Class. The file is Lena in gray-scale.
  - Load an image (using **cv::imread**)
  - Perform the sharpening filtering for the image by using a Laplacian filter, as you learn in this lecture (see Fig. 3.40 in page 39 in lecture 08).
  - Display the Laplacian image and the sharpened result image in OpenCV windows (using **cv::imshow**)

  - Try to use the blurred image as an input, which you have submitted for HW#1. Then perform the Sharping filtering using Laplacian filter.

- **Laplacian Filter**
  - Second derivatives

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y) \qquad \frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$



0	1	0	1	1	1
1	−4	1	1	−8	1
0	1	0	1	1	1
0	−1	0	−1	−1	−1
−1	4	−1	−1	8	−1
0	−1	0	−1	−1	−1

a b
c d

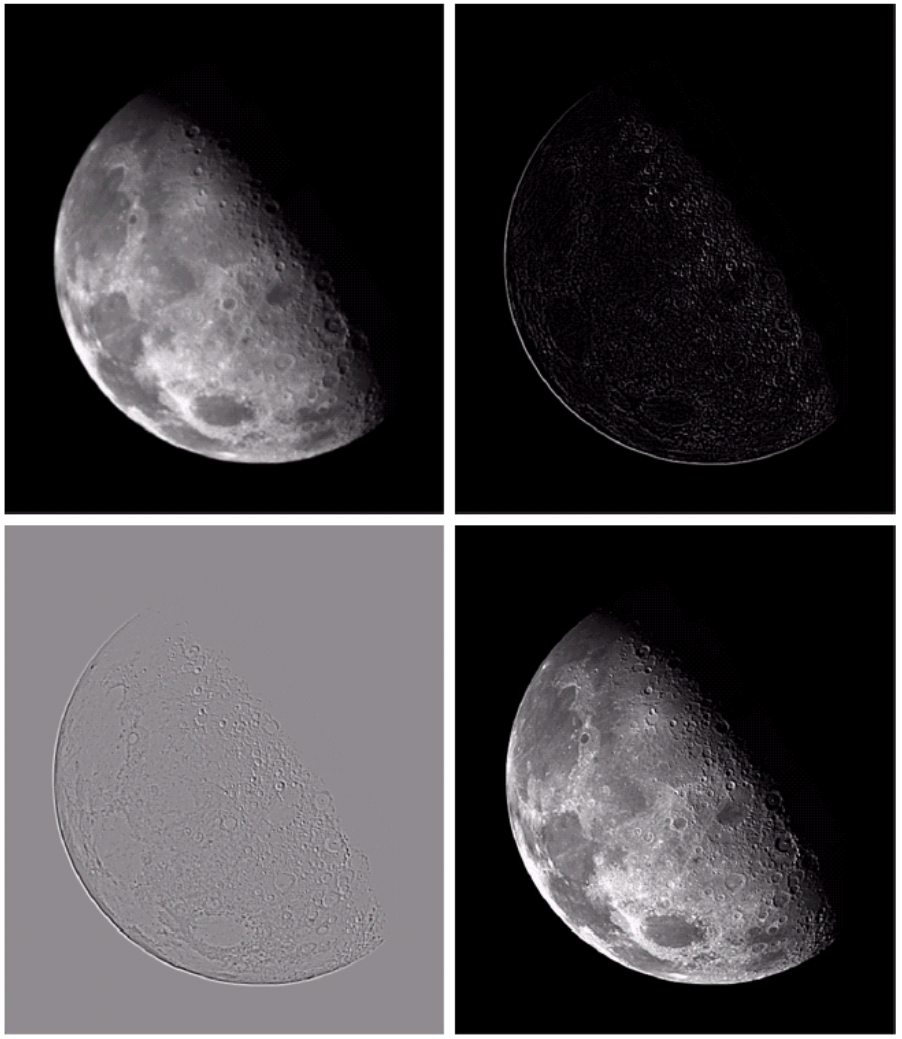
FIGURE 3.39
(a) Filter mask used to implement the digital Laplacian, as defined in Eq. (3.7-4). (b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian.

Sharpness Enhancement using Laplacian Filter

$$g(x,y) = f(x,y) + \left|\nabla^2 f(x,y)\right|$$

a b
c d

FIGURE 3.40
(a) Image of the North Pole of the moon.
(b) Laplacian-filtered image.
(c) Laplacian image scaled for display purposes.
(d) Image enhanced by using Eq. (3.7-5).
(Original image courtesy of NASA.)



Tip

- Save/Load project properties
  - Click on “Add New Project Property Sheet”
    → name your new property sheet
  - Set OpenCV properties for your new property sheet
  - Click on Save button on the “Property Manager”

  - Then, at the next time, you just need to “Add Existing Property Sheet”