Basic Image Editor using Python

Harsh Pal
18d070012@iitb.ac.in
Electrical Dual (CSP)
IIT Bombay

Abstract**:** The Basic Image Editor application is a simple and easy to use python based application where basic image processing operations such as histogram equalization, gamma correction, log transform, image negative, blurring, sharpening and edge detection can be implemented and played with. The application provides an easy to use GUI built on PyQt5 to perform these operations on grayscale and colour images. The operations in the application are performed on the V channel of color image in HSV format. The application was tested on a set of images, which is included in this report along with the output obtained and steps followed. The outputs obtained were satisfactory with the limitations on techniques used. The results can be further improved by using better image processing techniques and code optimization.

Official-Github-Repository:
https://github.com/harshpaal/EE610-image-editor

1. Introduction

Basic Image Editor is a python based GUI application which implements basec image processing operations on grayscale and color images without using any existing image libraries. The image processing operations implemented are:

- Histogram equalization
- Gamma correction (gamma as user input)
- Log transform
- Blurring
- Sharpening
- Edge detection*

Histogram equalization is implemented using the cumulative density function of the image. Gamma correction, log transform and edge detection are implemented as simple mathematical operations. Sharpening is implemented as a spatial domain operation with a 3x3 laplacian filter whose extent of sharpening is varied by multiplying different constants with the filter. Edge detection is also computed using laplacian.

Additional utility features of the applications are:

- Open Image using dialog box
- Save Image using dialog box
- Undo last change
- Undo all changes
- View image histogram

The application uses OpenCV python library for reading/writing images and color space transformation, Numpy library for array operation, PyQt5 for GUI and Matplotlib for histogram plotting. The GUI is designed using Qt designer application for windows.

2. GUI Design

A. *Overall approach*

The tools used for developing the GUI:

- PyCharm Community Edition [1] as IDE
- Qt Designer [2] for easy to use GUI code development with drag and drop features

First of all the basic layout of the application was designed in the Qt designer.

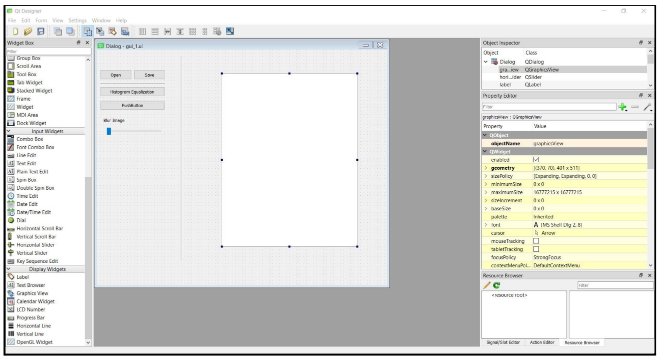


Fig. 2.0 Qt Designer window for GUI layout

After the final layout, the designer generates a .ui file which is converted into a .py file using pyuic5** in the Pycharm CE terminal. The obtained .py file is edited accordingly to be used as per the desired features of the Basic Image Editor.

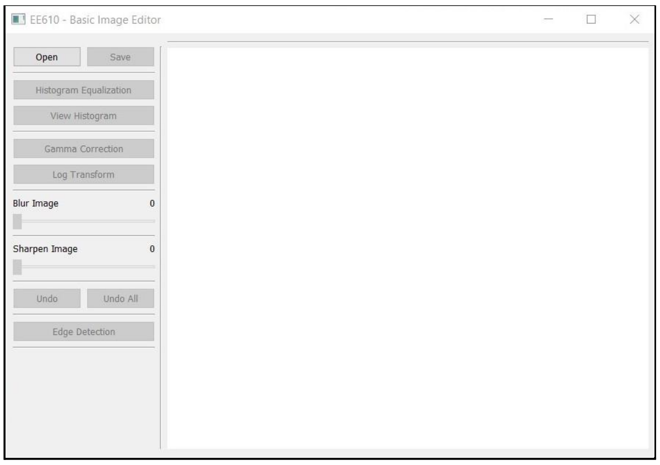


Fig. 2.1 Final GUI of Basic Image Editor

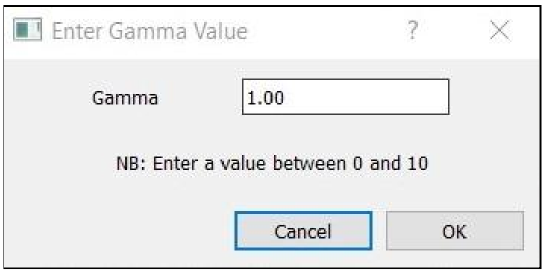
B. *GUI features*

The Basic Image Editor allows the user to open an image from the remote location through dialog box through Open push button and carry out desired image

* : *Additional feature for the assignment* ** : *Convert pyQt UI to python - Stack Overflow*

processing. Users can save the edited image through the Save push button. Undo push button reverts the last operation on the image and Undo All button reverts all the operations and returns the original image.
Major features of the application are the image processing push buttons which are as follows:

- Histogram equalization: this button performs histogram equalization of the image
- Gamma correction: this button performs gamma correction for range 0 to 10 after taking user input of the gamma value when pushed (Fig. 2.3)
- Log transform: this buttons performs log transformation with base 2
- Blur Image: this slider performs blurring of the image
- Sharpen Image: this slider performs sharpening of the image
- Edge Detection: this buttons performs edge detection on the image



Gamma 1.00

NB: Enter a value between 0 and 10

Fig. 2.3 Gamma correction

3. Image processing operations

A. Histogram equalization

Histogram equalization is used to uniformly distribute the image pixels over the entire intensity range, i.e. 0-255, thereby improving image contrast and thus reveals hidden features. The application uses cumulative distribution function(cdf) for the implementation of histogram equalization. For a completely equalized image, the cdf will be a straight line. By appropriate mapping of intensities from input image to equalized image the application attempts to approximately generate a straight cdf.
Steps followed by the application are:

- Compute image histogram of input image and setting range of intensities to be 0-255
- Calculating pmf
- Computing cdf from pmf
- Multiplying cdf by 255(=cdfn)
- Updating pixels in original image with intensities in cdfn

$$s = T(r) = (L - 1)\int_0^r p_r(w)dw$$ ....(intensity mapping)

$$p_s(s) = 1/(L - 1)$$ .....(cdf of equalized image)

B. Gamma correction

Gamma correction selectively enhances the contrast of pixels in a range of intensities depending on the value of gamma. For $\gamma > 1$, the contrast of darker pixels is improved hence contrast enhancement of bright looking images while for $\gamma < 1(>0)$ the contrast of lighter pixels is improved hence contrast enhancement of dark looking images.
The application implements gamma correction based on following equation:

$$s = Cr^{\gamma}$$

$$C = 255/255^{\gamma}$$

The constant C is chosen as above because at maximum intensity, mapped intensity will also be maximum.

C. Log transform

Log transform enhances the contrast of the lighter pixels in the image. The application implements log transform based on the following equation:

$$s = C\,log(1 + r)$$
$$C = 255 / log_2(256)$$

D. Blurring Image

Blurring smoothens the sharp transitions in the image intensity and is used for reducing noise and as a pre-processing task for other operations like edge detection. The application implements blur operation in spatial domain, by computing the mean of pixels in a window. The extent of blur is increased by increasing the window size. For color images, the filtering is done in the V channel of HSV channels.
Steps followed by the application are:

- Convert image from BGR to HSV format
- Define window as a square matrix of ones
- Perform convolution of window and V channel of image (zero padding was used to handle edge pixels)
- Normalize the pixel values by dividing by the sum of element in the window

E. Sharpening Image

Sharpening highlights the intensity transitions in the image. It can be implemented by using spatial filtering techniques (sobel, laplacian, Gaussian, etc.) or by frequency domain operations like high pass filtering. The application implements sharpen operation in spatial domain, by computing Laplacian of the image over a 3x3 window. The extent of sharpening is controlled by varying the constant used with Laplacian. For color images, the filtering is done in the V channel of HSV channels.
Steps followed by the application are:

- Convert image from BGR to HSV format

- Define Laplacian window as :

  W = [1 1 1, 1 − 8 1, 1 1 1]
- Perform convolution of window and V channel of image to obtain Laplacian(zero padding in the edge)
- Implement following operation on image:

$$\boxed{g(x,\ y)\ =\ f(x,\ y)\ +\ C[\nabla^2 f(x,\ y)]}$$

- Here constant C is varied between 0 and 1 to modify the extent of sharpening
- The pixel values outside the range 0 and 255 is mapped to 0 (values < 0) or 255 (values > 255) as required

F. *Edge Detection*

Edge detection detects the intensity transitions in the image, which translates as edges in the image. The application implements edge detection by computing Laplacian of the image over a 3x3 window.

Steps followed by the application are:

- Convert image from BGR to HSV format
- Define Laplacian window as :

  W = [1 1 1, 1 − 8 1, 1 1 1]
- Perform convolution of window and V channel of image to obtain Laplacian(zero padding in the edge)
- The pixel values outside the range 0 and 255 in edge image is mapped to 0 (values < 0) or 255 (values > 255) as required

4. Experiments and results

A. *Experiments with Histogram equalization*

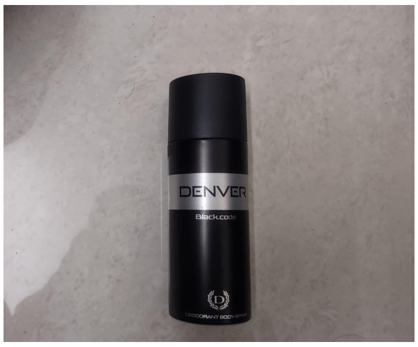


Fig. 4.0.0 Original Image

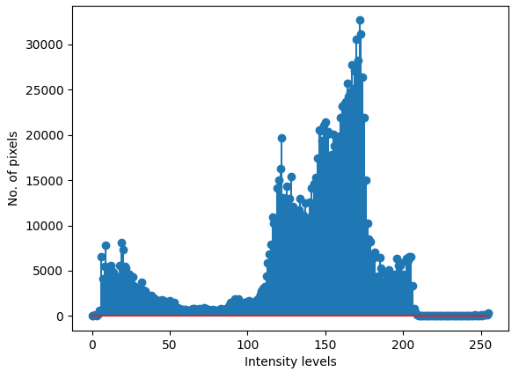


Fig. 4.0.1 Original Histogram

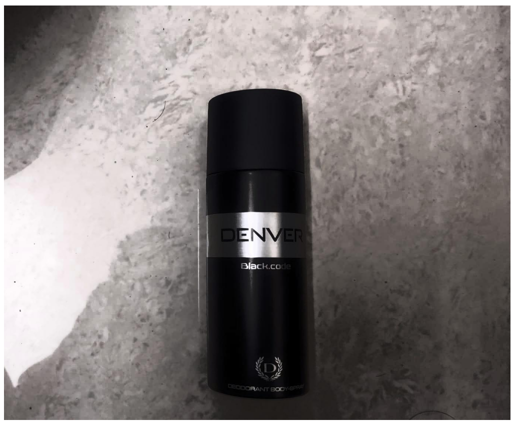


Fig. 4.0.2 Equalized Image

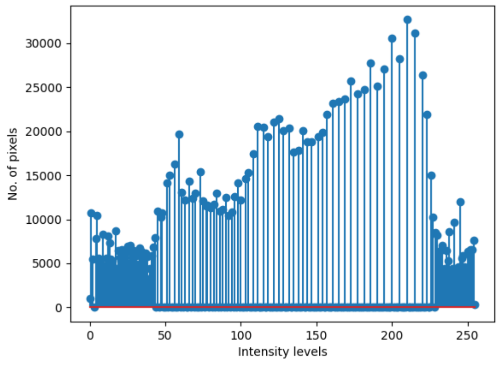


Fig. 4.0.3 Equalized Histogram

Results of the experiment:

- Original input image(Fig. 4.0.0) was chosen with only extreme intensities so that we can get peaks in the original histogram(Fig. 4.0.1)
- Original histogram has two peaks
- The equalized image(Fig 4.0.2) has more lower intensities and lesser higher intensities than original image
- Equalized histogram (Fig. 4.0.3) is spread out, though it is not flat as expected

B. *Experiments with Gamma correction*

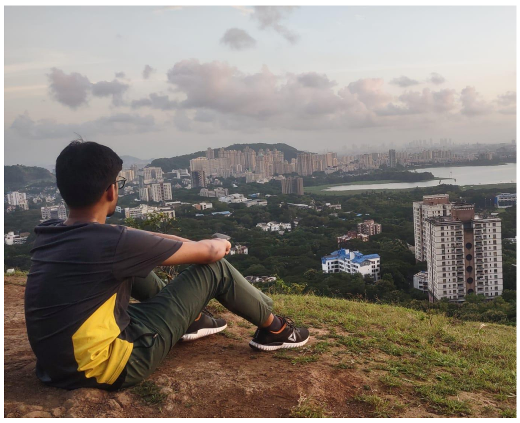


Fig. 4.1.0 Original Image

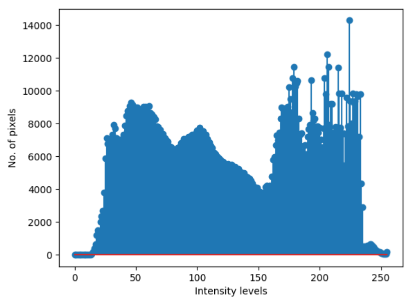


Fig. 4.1.1 Original Histogram

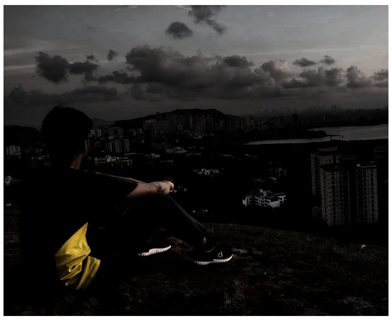


Fig. 4.1.2 Gamma corrected Image (γ = 5)

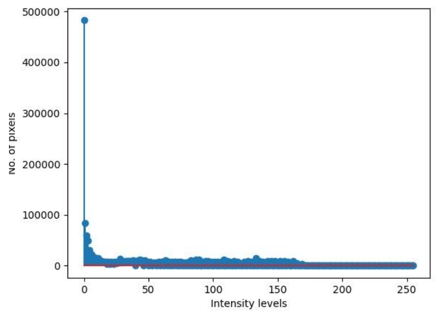


Fig. 4.1.3 Histogram of gamma corrected image (γ = 5)

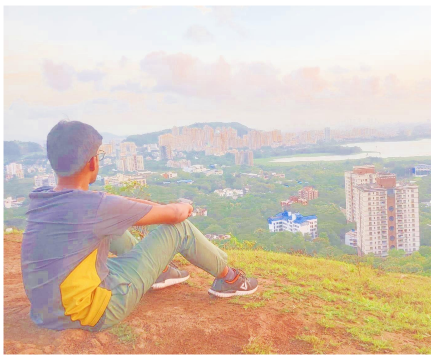


Fig. 4.1.4 Gamma corrected image (γ = 0.2)

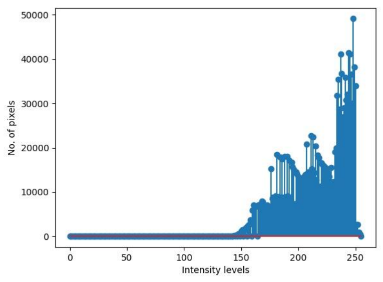


Fig. 4.1.4 Histogram of gamma corrected image (γ = 0.2)

Results of the experiment:

- Original Input image(Fig. 4.1.0) was chosen to have intensities from all the levels and hence a flat histogram(Fig. 4.1.1). So an image with natural elements seemed appropriate.
- With γ > 1 (5 in our case), the contrast of low intensities are improved, hence we get relative darker image(Fig. 4.1.2)
- With γ < 1 (0.2 in our case), the contrast of high intensities are improved, hence we get relatively brighter image with more bright colors(Fig. 4.1.4)

C. Experiments with Log transform

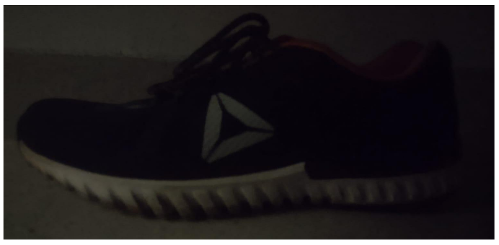


Fig. 4.2.0 Original Image

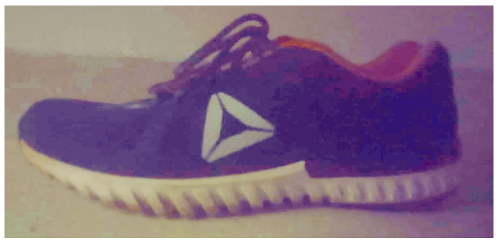


Fig. 4.2.1 Log transformed Image

Results of the experiment:

- Original Image (Fig. 4.2.0) was clicked intentionally in low illumination inorder to have low intensities in the image
- Performing log transform enhanced the details in the transformed image(Fig. 4.2.1) which were slightly hidden in the original due to low intensity
- Details like shoe laces and Inside orange color are distinguishable in log transformed image

D. Experiments with Image Blurring

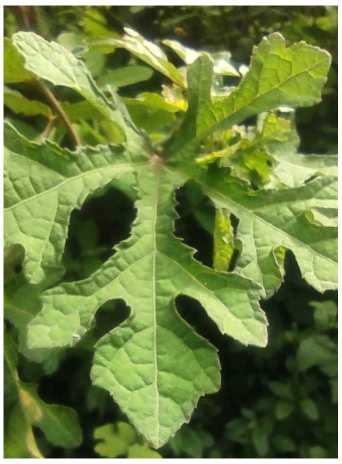


Fig. 4.3.1 Original Image

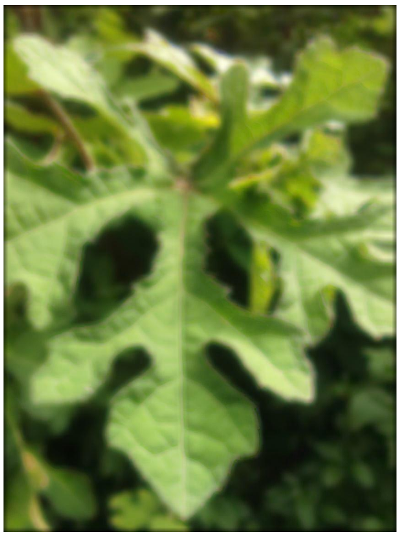


Fig. 4.3.2 Blurred Image

Results of the experiment:

- Original Image(Fig. 4.3.1) was chosen to have some edges whose visibility can be manipulated with blurring
- Blurred Image(Fig. 4.3.2) has less visible veins in leaves then the original image

E. Experiments with Image Sharpening

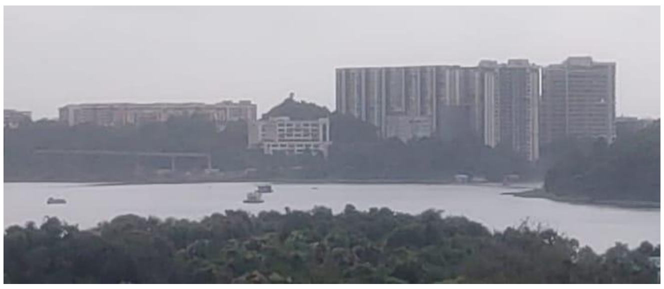


Fig. 4.4.1 Original Image

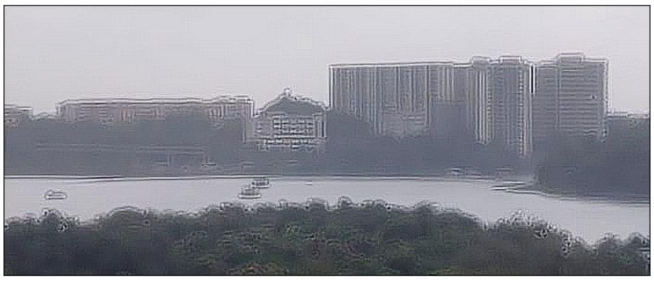


Fig. 4.4.2 Sharpened Image

Results of the experiment:

- Original Image(Fig. 4.4.1) was chosen to have some edges which can be enhanced after sharpening
- Sharpened Image(Fig. 4.4.2) has enhanced edges, though some are over enhanced and thus overflowing

F. Experiments with Edge Detection

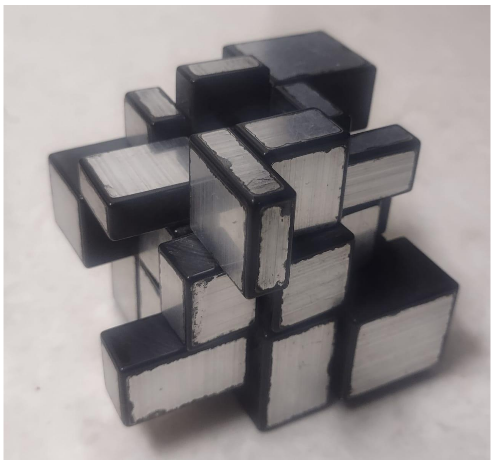


Fig. 4.5.1 Original Image

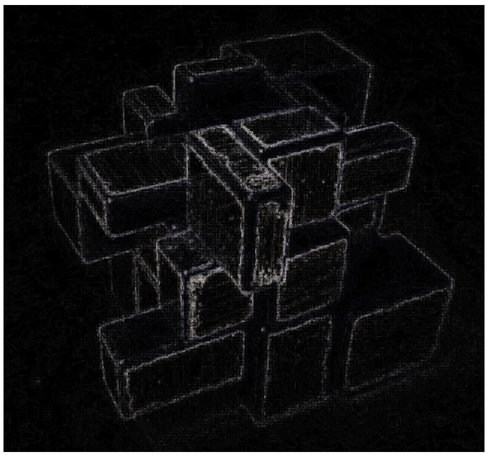


Fig. 4.5.2 Edge detection

Results of the experiment:

- Original Image (Fig. 4.5.1) was chosen to have distinct edges which can be easily detection in edge detection
- Edge detected Image (Fig. 4.5.2) have almost all the distinct edges distinguishable with naked eyes in the original image

5. Conclusion and discussion

A. Main challenges which were overcome

- Unfamiliarity with Qt Designer and application development in Python, which consumed a significant amount of initial time in the project due to surfing the internet and searching for appropriate resources.
- Resolving unwanted dark pixels in sharpened and log transform images also consumed a considerable amount of time. The reason was the range of pixel intensities and was solved with extensive googling

B. Possible future work

- Feature of image manipulation with directly tweaking the histogram with sliders
- Feature of AI based image editing with Generative Adversarial Networks(GAN) like Style transfer
- Implementation of DFT and FFT algorithms for image enhancement

6. References

[1] Gonzalez, Rafael C., and Woods, Richard E. "Digital Image Processing", Third Edition(2008)

[2] https://opencv.org/

[3] https://docs.scipy.org/doc/numpy/reference/

[4] https://pypi.org/project/PyQt5/

[5] Qt-Designer-Manual

[6] Intelligent-Coding-Assistance(jetbrains.com)

[7] GUI-application-using-python(github.com)