1. **Point processing transformations (25%)**

These are four-point processing transformations:

* Thresholding
* Gray-scaling
* Negative transformation
* Histogram Equalization

1. Please explain the basic concepts of above point processing transformations. Complete your explanations with the mathematical equations of the transformations.

**Answer:**

1. **Thresholding**

Thresholding creates binary images from grey-level ones by turning all pixels below some threshold to zero and all pixels about that threshold to one.

If g(x, y) is a threshold version of f(x, y) at some global threshold T

**g(x, y) = 1** **if f(x, y) ≥ T**

**0 otherwise**

1. **Gray-scaling**
2. **Logarithmic Transformations**

The general form of the log transformation is

**s = c \* log(1 + r)**

The log transformation **maps a narrow range of low input grey level values into a wider range of output values.**

**Usually set c to 1 .Grey levels must be in the range [0.0, 1.0]**.

1. **Power Law Transformation**

Power law transformations have the following form

**s = c \* r γ**

**Map a narrow range of dark input values into a wider range of output values or vice versa**. **Varying γ gives a whole family of curves**.

1. **Negative Transformation**

Negative images are useful for enhancing white or grey detail embedded in dark regions of an image. It basically works by reversing the intensities of the image points. The formula:

**s = maxintensity - r**

**s = (L – 1) – r**

Here **L will be taken as 256** and **256-1 =255 gives us the max intensity**. **r is the intensity at the point to be considered**.

So, what happens is that, **the lighter pixels become dark** and **the darker picture becomes light**. And it results in image negative.

1. **Histogram Equalization**

Histogram Equalization is a computer image processing technique **used to improve contrast in images**. It accomplishes this by effectively spreading out the most frequent intensity values, i.e. **stretching out the intensity range of the image**.

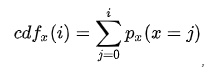
This method usually increases the global contrast of images when its usable data is represented by close contrast values. This **allows for areas of lower local contrast to gain a higher contrast**.

Consider a discrete grayscale image {x} and **let ni be the number of occurrences of Gray level i**. The probability of an occurrence of a pixel of level i in the image is



**L being the total number of gray levels in the image** (typically 256), **n being the total number of pixels in the image**, pixel value i, normalized to [0,1].

**Let define cumulative distribution function corresponding to px as**



**which is also the image's accumulated normalized histogram.**

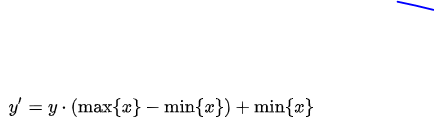
Would like to create a transformation of the form y = T(x) to produce a new image {y}, with a flat histogram. Such an image would have a linearized cumulative distribution function (CDF) across the value range.



for some constant K. The properties of the CDF allow to perform such a transform (see Inverse distribution function); it is defined as:



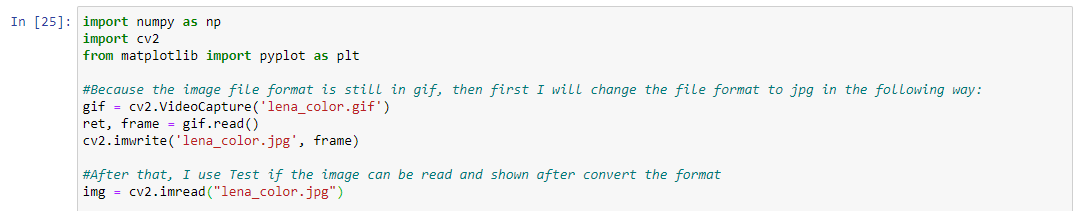
where k is in the range [0,L]). Notice that T maps the levels into the range [0,1], since used a normalized histogram of {x}. **To map the values back into their original range, the following simple transformation needs to be applied on the result**:



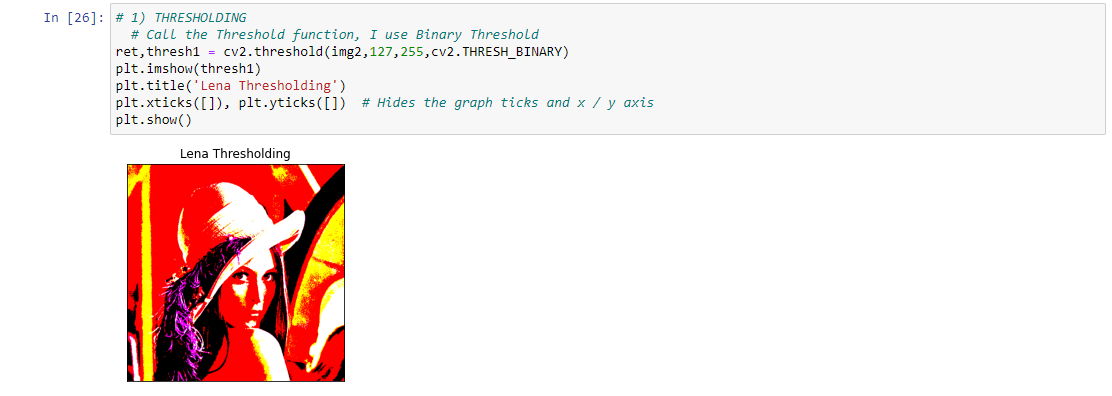
1. Furthermore, please download the image from this URL: https://qrgo.page.link/YNzeX The first step is you load the image (lena\_color.gif) and implement all above point processing transformations. It can be done by using OpenCV library.

**Answer:**

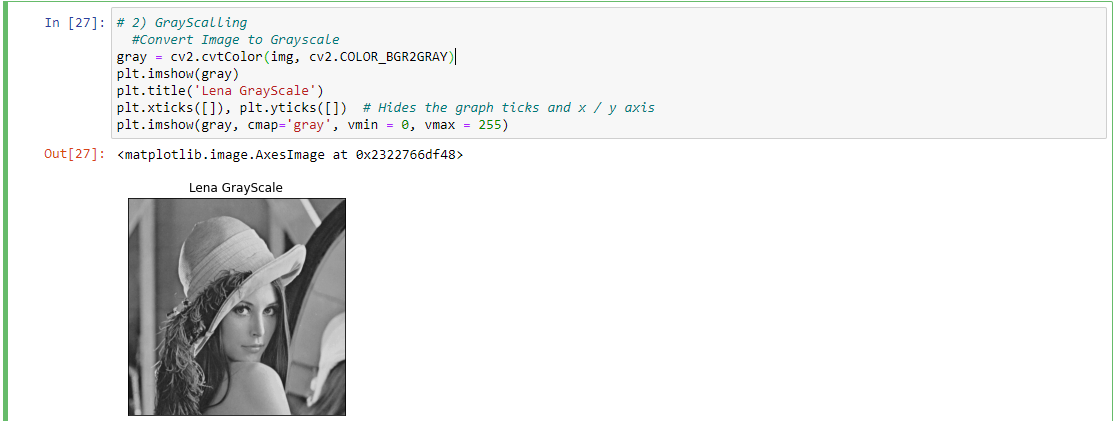
**Load The Image:**



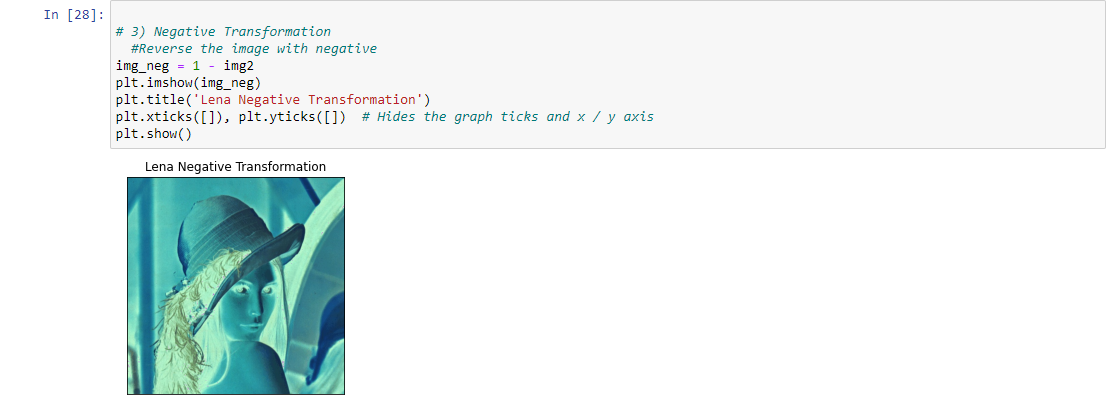
1. **Thresholding**



1. **Gray-scaling**

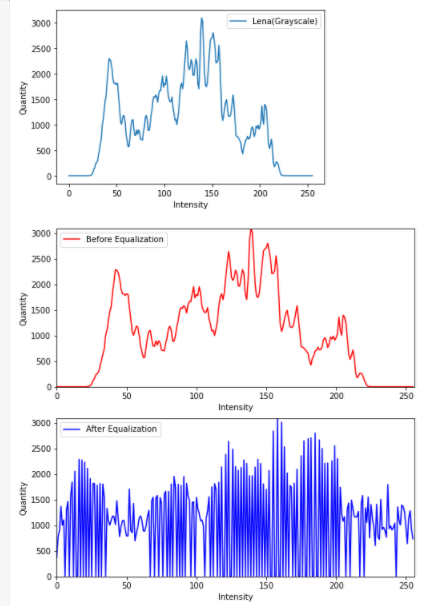


1. **Negative Transformation**



1. **Histogram Equalization**





1. **Convolution (25%)**

Please download the image from this URL: https://qrgo.page.link/Vt7Ag The first step is you load the image (opera\_house.jpg) and convert it into grayscale. It can be done by using Pillow library. And the two 3x3 kernels

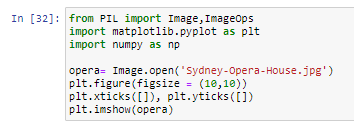


Please find the solution for: Convolution operation without border padding

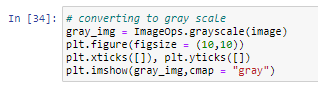
1. Implement the above calculations using Numpy library.

**Answer:**

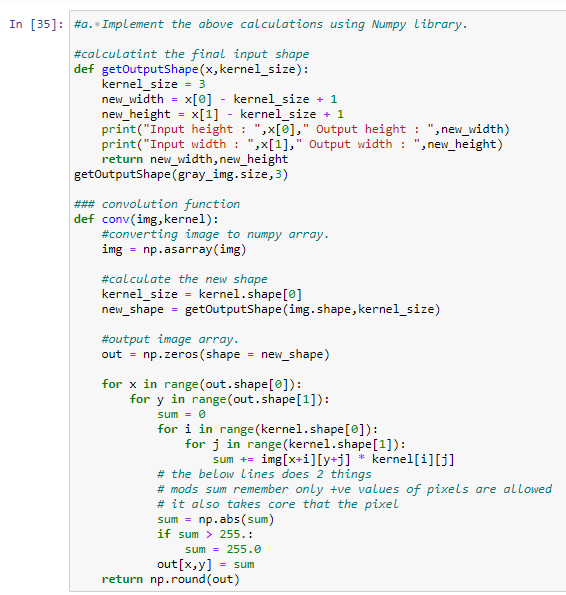
**Load Image:**



**Convert to Grayscale:**

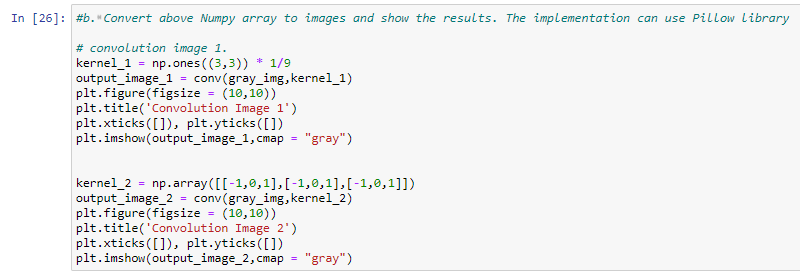


**Implementation of Convolution with 3x3 kernels:**



1. Convert above Numpy array to images and show the results. The implementation can use Pillow library.

**Answer:**





1. **Harris corner detector (25%)**

Please answer below questions related to Harris corner detection as follow:

* 1. Explain the basic idea to detect the Corner Points in an image

**Answer:**

**Corners are mainly formed by the combination of two or more edges**. These corners may or may not define the boundary of an image. **Harris corner detection algorithm is found out by calculating each pixel’s gradient**. **If the absolute gradient values in both the directions are great, then consider the pixel as a corner**.

**This method is followed by mathematical formulas**. By taking into considerations of the differential of the corner score with respect to direction directly, **Harris and Stephens improved the Moravec's corner detector instead of using shifted patches**.

**The accuracy of the extracted points will affect the accuracy of camera calibration parameters and its post processing problems of image quality**.

* 1. If M is second moment matrix of image derivatives, please explain mathematically how to classify image points based on M Matrix!

**Answer:**

* **Step 1 (Defines the Derivatives):**

**Ix = Iy = IxIy =**

* **Step 2 (Find out the product of derivatives of each pixel):**

**Ix2 = Ix X Ix Iy2 = Iy X Iy Ixy = Ix X Iy**

* **Step 3 (Calculate the covariance matrix M):**

**M = ∑ W(x,y)**

**xy**

**W(x,y) is calculated by a windows function, that is:**

**xy**

**E(u,v) = ∑ W(x,y) [I(x+u,y+v) - I(x,y)]**

**Where:**

**I(x+u,y+v) is known as Shifted Intensity**

**I(x,y) is known as the Intensity**

**The Above Equation can be simplified as:**

**E(u,v) ≈ M**

**From that, we can conclude that:**

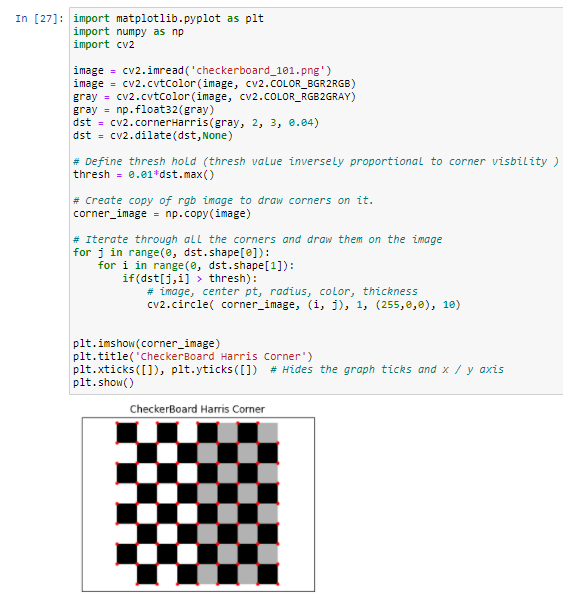
**M = E(u,v)**

* 1. Harris corner detector algorithm is based on above idea, together with additional step to enhance the detector. Please explain each step of Harris corner detector based on its original paper: C.Harris and M.Stephens. "A Combined Corner and Edge Detector“, Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988

**Answer:**

* **Step 1 (Convert to grayscale from the original image)**
  + - * + **because we need RGB channel for corner detection**, Moreover, it **reduces the size of information in the image** and it **helps to make the process fast**.
* **Step 2 (Apply Sobel operator)**
  + - * + **To find the x and y gradient values for every pixel** in the grayscale image.
* **Step 3(Calculate Harris Value)**
  + - * + **Each pixel in the grayscale image**, compute corner strength function.
* **Step 4 (Find all pixels that exceed a certain threshold and are the local maxima within a certain window)**
  + - * + **To prevent redundant dupes of features**.
* **Step 5 (Compute Feature Descriptor)**
  + - * + **Each pixel that meets the criteria in Step 5**, compute feature descriptor.
  1. Please download the image from this URL: https://qrgo.page.link/jNBDK The first step is you load the image (checkerboard\_101.png) and implement Harris corner detector. It can be done by using OpenCV library.

**Answer**:



1. **Canny edge detector (25%)**

Canny edge detector was developed by John F. Canny in 1986 and uses a multi-stage algorithm to detect multiple edges in an image. Briefly describe the procedures involved in each stage, implement and apply it to this image from this URL: https://qrgo.page.link/HVHsH

* 1. Noise Reduction. For this purpose, what kind of filter is used? And why is this stage very important to do in the first place? Explain it!

**Answer:**

**What kind of filter:**

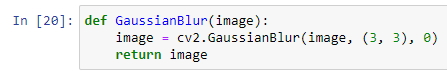
I am using a **Gaussian filter**.

**Why this stage very important:**

The edge detection algorithm is very sensitive to image noise. As a result, it is essential to filter it to prevent detecting false.

**Implementation:**

* **Apply Gaussian Blurring**



* **Result**



* 1. Finding Intensity Gradient of the Image. What does this stage produce? And what filters do you use to achieve it?

**Answer:**

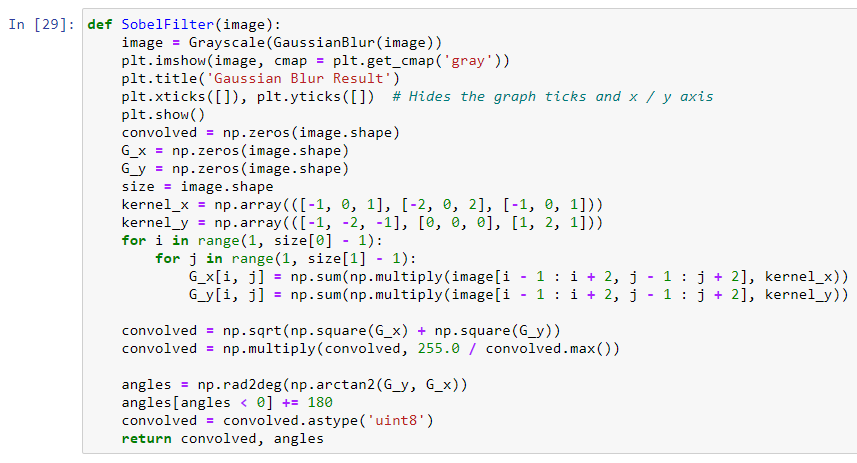
**The Stages Produce:**

To detect the change of pixels' intensity on both X-axis and Y-axis The gradient detects the direction and the intensity of the edge by using edge-detection operators.

**What kind of filter:**

I used the **Sobel Filter**.

**Implementation:**



**Result:**



* 1. Non-maximum Suppression. What is meant by this stage?

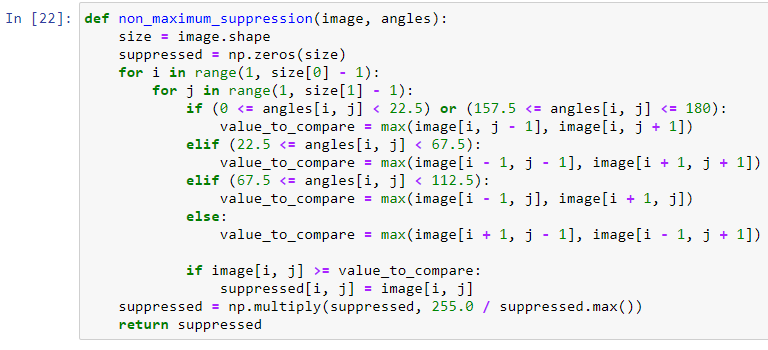
**Answer:**

**Meant by this stage:**

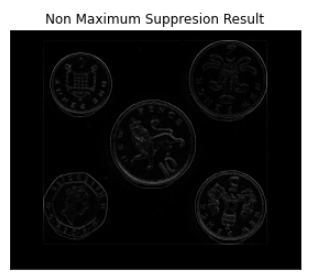
Is thinning technique. Once the edge direction is calculated we can trace them in the image. When looking at a pixel there are only four directions when checking the surrounding pixels in a reverse trigonometric direction:

* 0 degrees => horizontal direction
* 45 degrees => positive diagonal
* 90 degrees => vertical direction
* 135 degrees => negative diagonal

**Implementation:**



**Result:**



* 1. Hysteresis Thresholding. What is the purpose of this stage? How do you perform double thresholding to achieve this goal?

**Answer:**

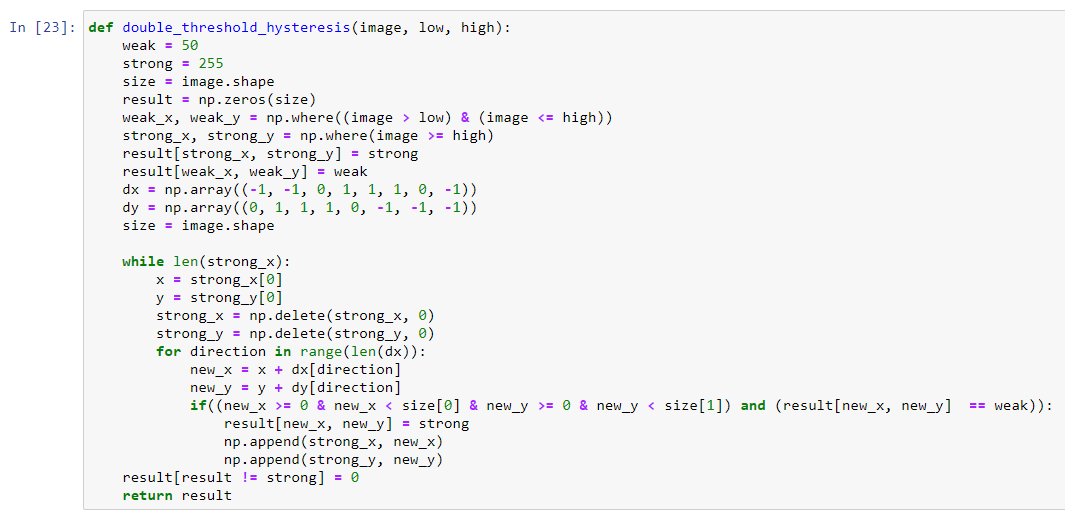
**How I Perform Double Thresholding:**

I am filtering the image. This is **accomplished by sorting all pixels into 2 categories (weak and strong)**. **If a pixel's value is greater than our high threshold, we consider it a strong one**. **If the pixel is brighter than the low threshold but darker than the high threshold, we consider the pixel to be weak**. **Pixels that are smaller than the low threshold are being ignored**.

**Purpose of Edge tracking by hysteresis:**

Now our image consists of weak pixels and strong pixels. We **must make all true edge pixels strong**( **even if now they are weak**) and **all the non-edge pixels should be transformed into zeros**. A weak pixel is transformed into a strong one only if it has a strong neighbour on any of the 8 directions.

**Implementation:**



**Result:**

