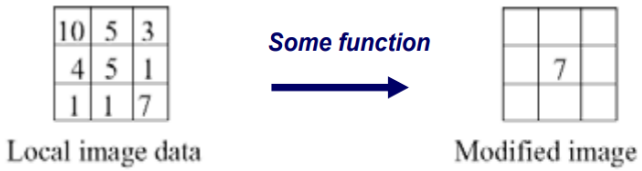
IMAGE FILTERING AND EDGE DETECTION

Prepared by team9:

1. Rawan Abdelhameed
2. Mostafa Mahmoud
3. Esraa Ali
4. Omar Mostafa
5. Yahia Saeed

1. Introduction

- Image filtering: is to modify the pixels in an image based on some function of a local neighborhood of the pixels.



- Edge detection:
  - I. Goal: Identify sudden changes (discontinuities) in an image (Most semantic and shape information from the image can be encoded in the edges).
  - II. Ideal: artist's line drawing (but the artist is also using object-level knowledge).

2. Methods

This project was mainly implemented using the following python packages:

- Numpy : for computations
- Matplotlib : for visualization and plotting.
- Streamlit : for build our app.
- CV2 : For reading and handling uploaded image.
- **OS :** Python OS module provides the facility to establish the interaction between the user and the operating system.
- PIL : for dealing with images.

3. Algorithms:

1. Fourier transform:

   Parameter: image.

   This function used to apply Fourier transform for images as np.fft.fft2 used to calculate Fourier and np.fft.fftshift used to Shift the zero-frequency component to the center of the spectrum.

   Return: image in Fourier domain.

2. Apply LPF:

   Parameters: image, cut off frequency as an input from the user.

   This function used to apply low pass filter in frequency domain, calculate Fourier for an image, get the dimensions of the image using. Shape function, create zeros map with the same dimensions of the image then looping the original image dimensions and compare it with the cutoff frequency.

If it little than or equal cut of frequency -> set this px in the mask as 1 else set this px in the mask as 0 then multiply the mask on the Fourier of the image and the final step to reconstruct the image back.

Return: image in frequency domain, filtered image, and
low pass mask map.

3. Apply HPF:

   Parameters: image, cutoff frequency as input from the user

   First, we call the function of low pass filter, calculate the highpass mask as lowpass mask, multiply the high pass mask and image in Fourier domain, and reconstruct the image back.

   Return: None.

4. Apply gaussian noise:

   Parameters: image, mean, and sigma.

   The function calculates the width and height of the image to apply the noise and when sigma is changed the noise changes alternatively.

   Return: noised image.

5. Apply uniform noise:

   Parameters: image, and noise value.

   This function calculates the image width and height to apply noise. we create a uniform distribution whose lower and upper bounds are the minimum and maximum pixel values (0 and 255 respectively) along the dimensions of the image.

Return: uniform noisy image.

6. Apply salt and pepper noise:

   Parameters: number of black and white pixels, and image.

   The function Gets the dimensions of the image then adds black and white pixels to the image randomly and also allows the user to control the number of the pixels of the noise.

7. Apply average filter:

   Parameters: image, and kernel size.

   This function calculates the image width and height, and also calculates the mask to apply the filter.

   Return: average filtered image.

8. Apply median filter:

   Parameter: image.

   This function calculates the image width and height to obtain the number of rows and columns of the image then find the median of the pixels and replace he center pixel by the median.

9. Apply gaussian filter:

   Parameter: image path, and sigma.

   The function converts the image into numpy array and apply the kernel of size 9x9.

   Return: gaussian filtered image.

10. Make_gaussian:

    Parameters: standard deviations valeral, size of output kernel

    This function used to create gaussian kernel to used it in create hybrid image.

    Return: kernel with size of (size*size).

11. Gaussian:

    Parameters: distance square from center of gaussian distribution, standard deviation.

    This function used to sample one instance from gaussian distribution.

    Return: a sampled number obtained from Gaussian.

12. Apply hybrid filter:

    Parameters: first image, second image, cutoff frequency of lowpass filter, cutoff frequency of high pass filter.

    This function used to create hybrid image by calculating the mask of both high pass and lowpass the apply padding on both of them then multiply each mask and the Fourier of the selected image and finally reconstruct the addition of the result.

    Return: None

13. Thresholding:

    Thresholding of an image aims to separate an object in the image (foreground) from the background using a threshold value. Pixel values that are higher than the threshold is given the value 255 (white) and ones that are lower are given the value 0 (black). Thresholding is

performed either globally or locally. In global thresholding, we apply one threshold value on all the image pixels. In local thresholding, we partition the image into four sections, where we apply a unique threshold value for each section.

The threshold value is either given manually by the user, or is calculated using Otsu's method. Otsu's method involves iterating through all the possible threshold values and calculating the weight and mean to find the between class variance. The threshold value is the one where the between class variance is the maximum.

14. RGB Histogram:

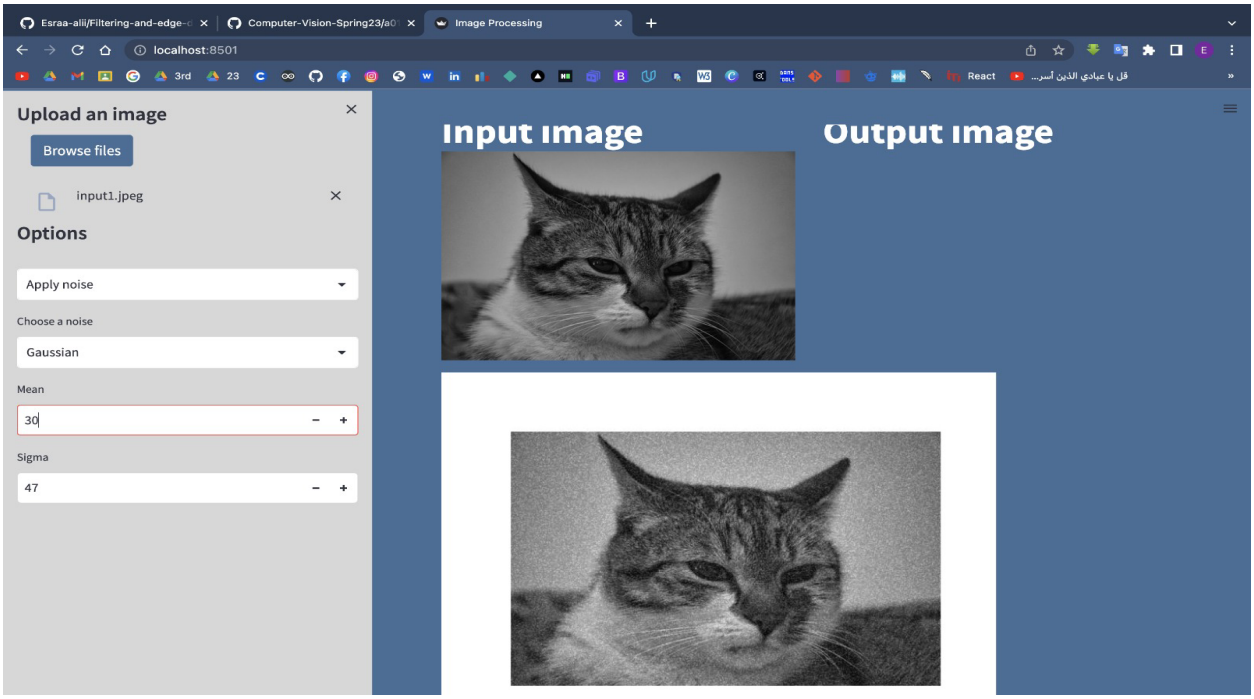
    We separate each color channel in the image into different arrays, convert the output arrays from 2D to 1D, then use them to draw their histograms, distribution and cumulative curves.

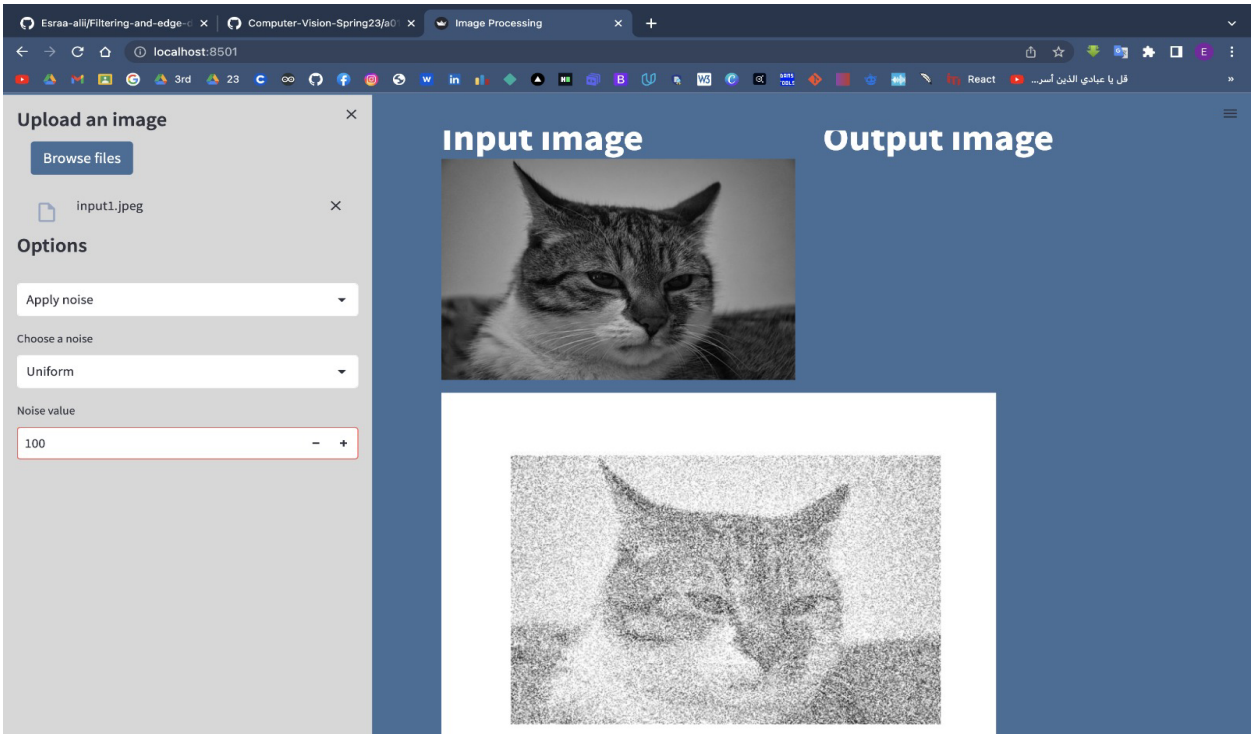
    In the equalized mode, we calculate the probability distribution and the cumulative distribution of the pixel values frequencies from the histogram, then perform linear interpolation to get the new values to plot the histogram, distribution and cumulative curves.

15. Equalize the image by:
    - convert the image to grayscale.
    - convert it to a numpy array.
    - normalized cumulative histogram.
    - mapping pixel lookup table.
    - transform pixel values to equalize.
    - reshape and write back into the image's array.

Our project can be divided into 10 main points as follows:

- Upload an image and add noise to this image (uniform, salt& pepper, and Gaussian noise).





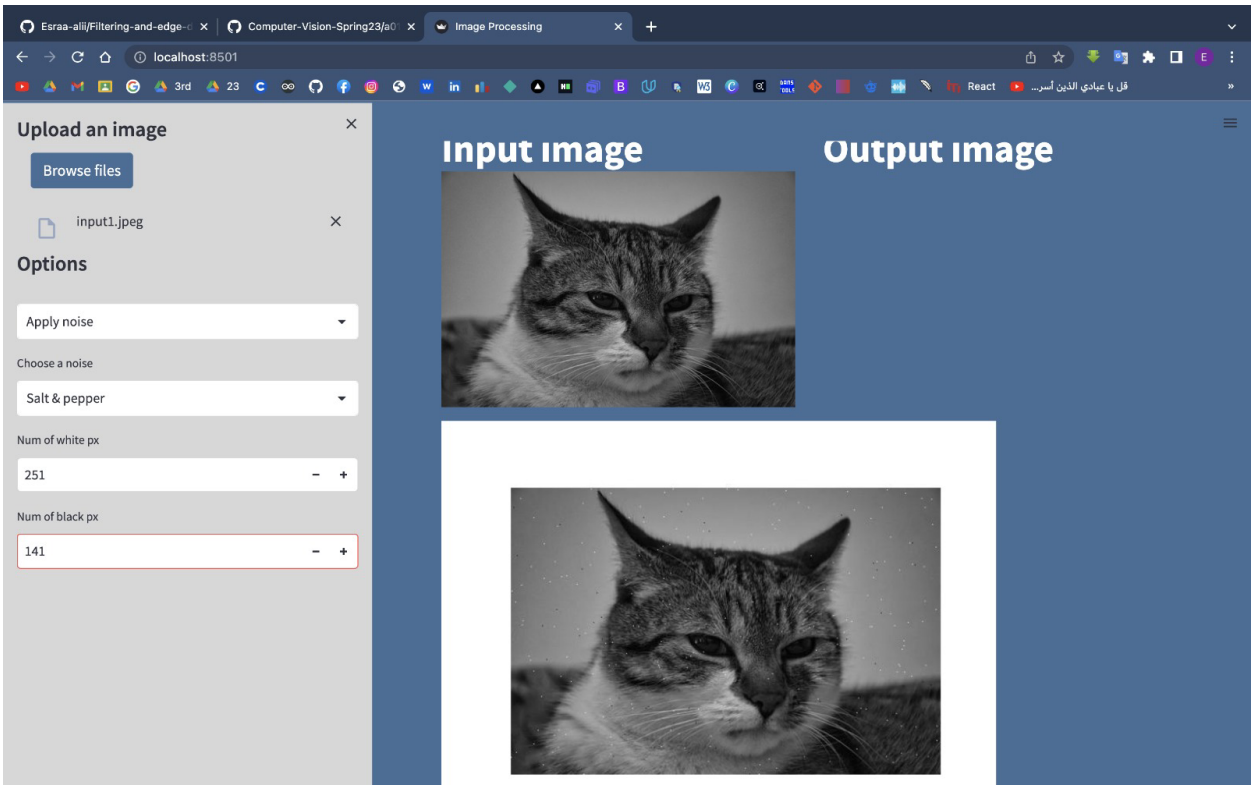
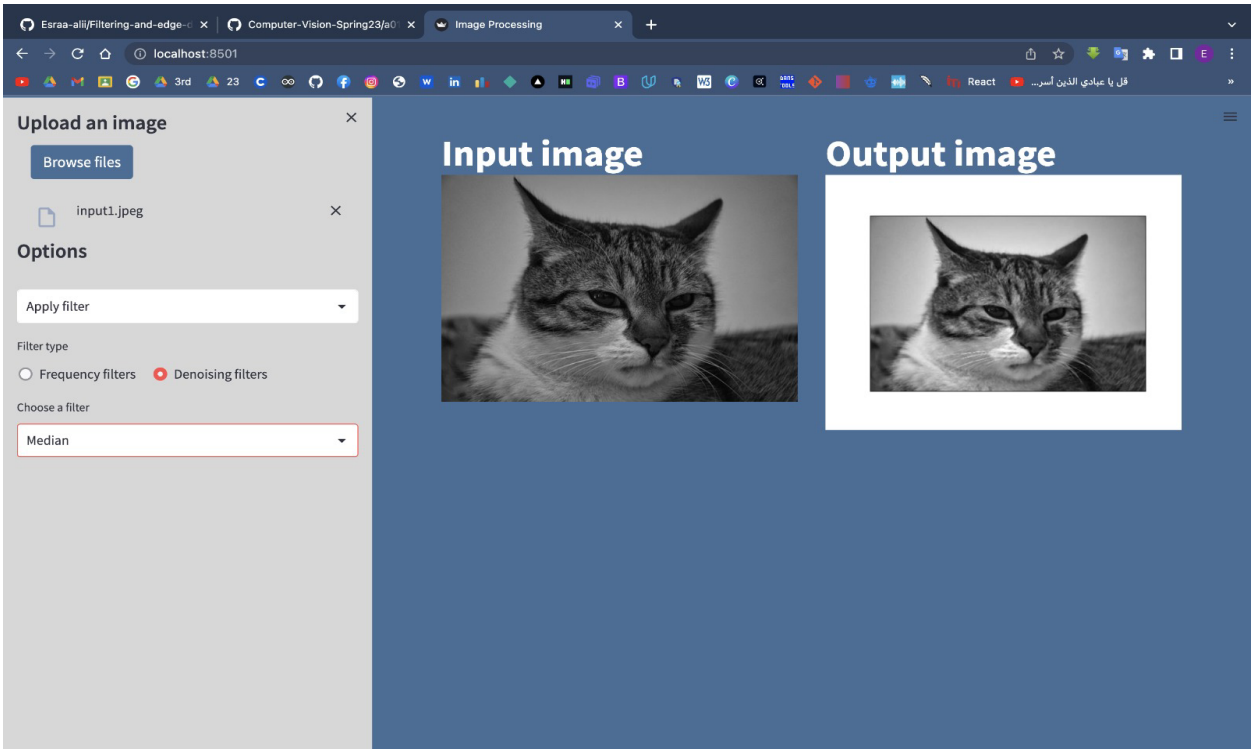
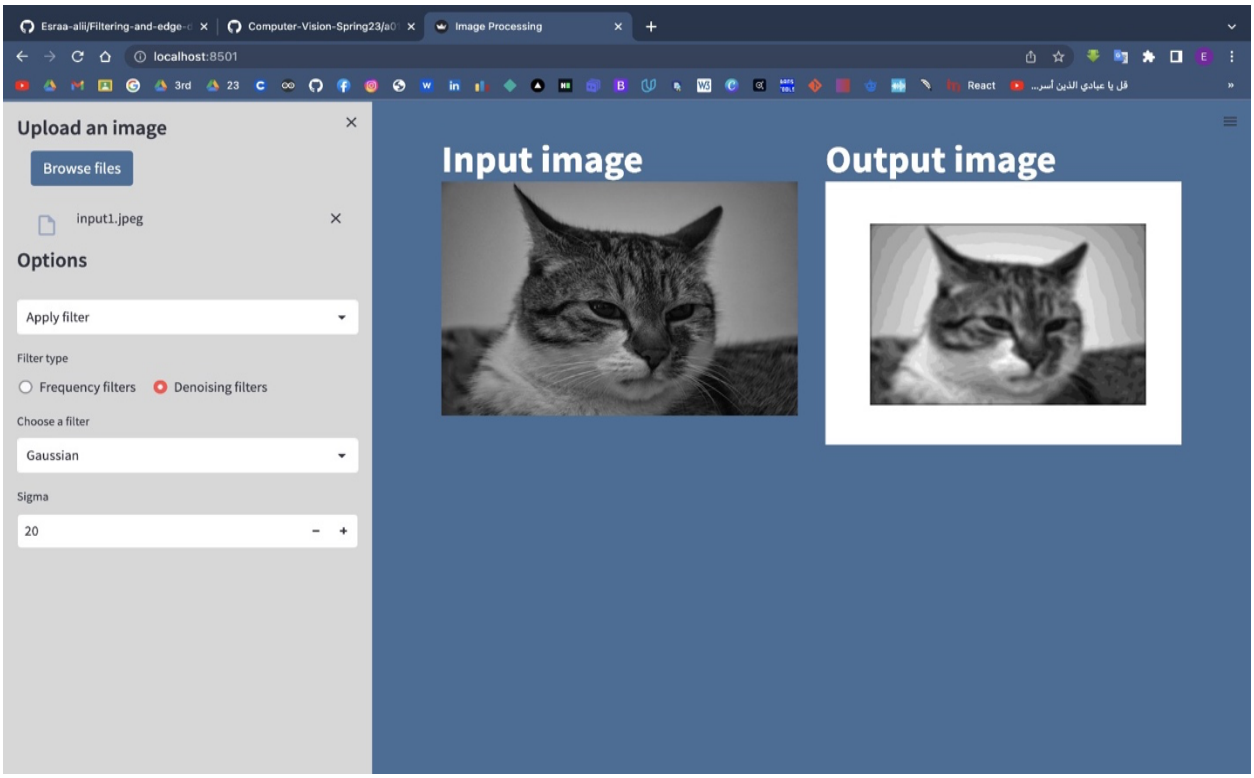
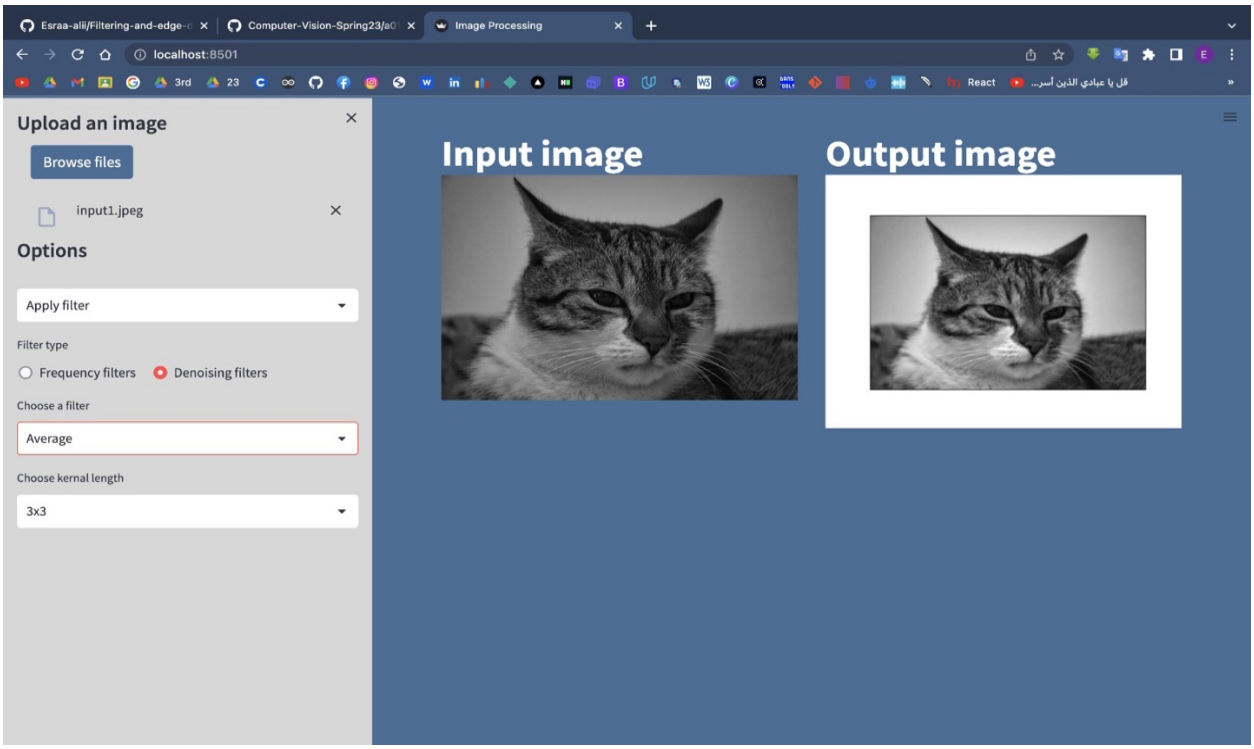


Image Processing
localhost:8501
Upload an image
Browse files
input1.jpeg
Options
Apply noise
Choose a noise
Salt & pepper
Num of white px
251
Num of black px
141
Input Image
Output Image

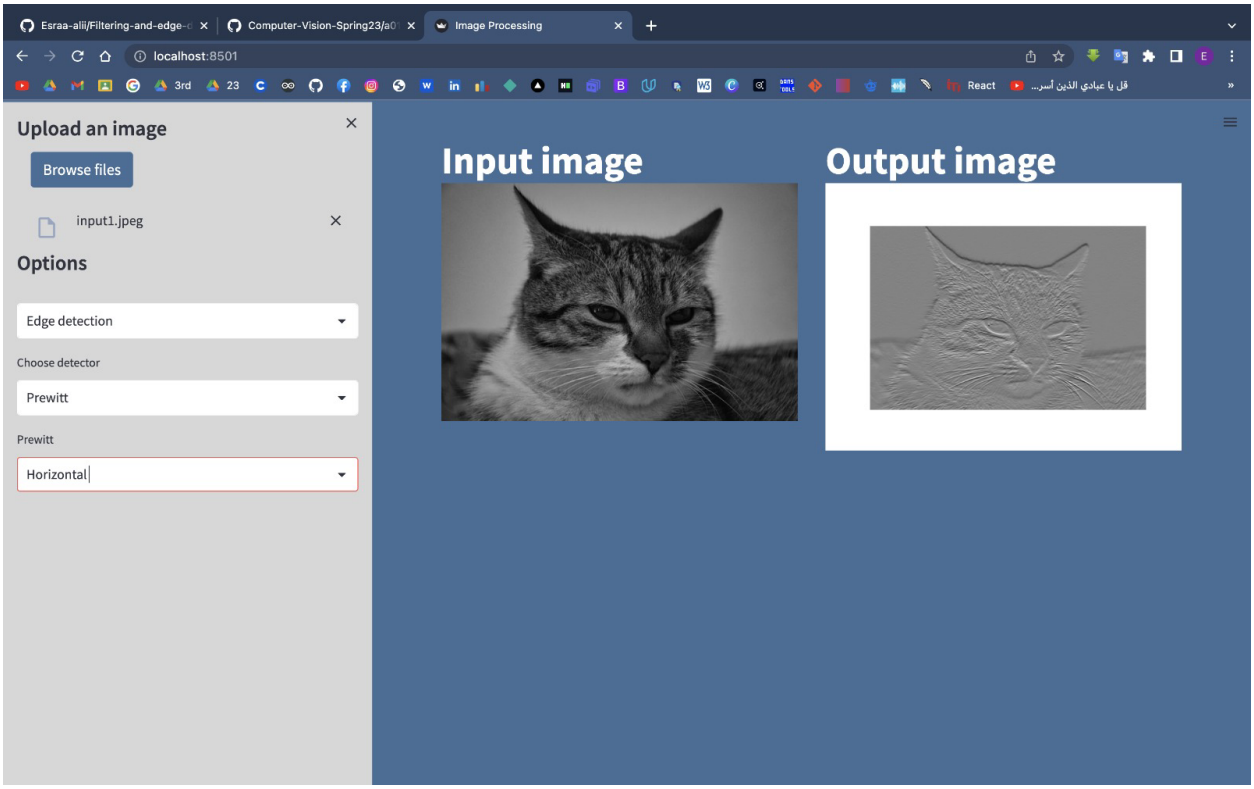
- Filter the noisy image using low pass filters (average, gaussian, and median filters).







- Detect edges in the image using Sobel, Robert, Prewitt, and Canny detector.



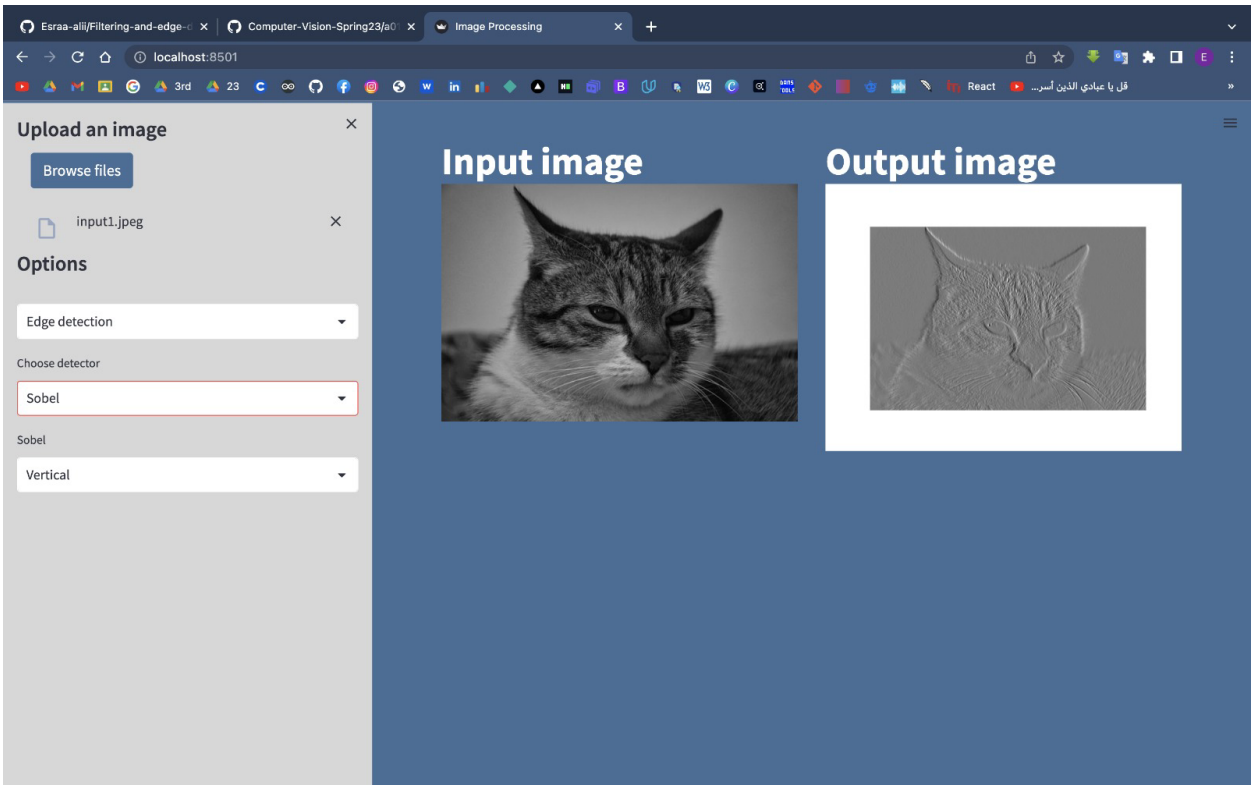


Image Processing
localhost:8501
Upload an image
Browse files
input1.jpeg
Options
Edge detection
Choose detector
Sobel
Sobel
Vertical
Input image
Output image

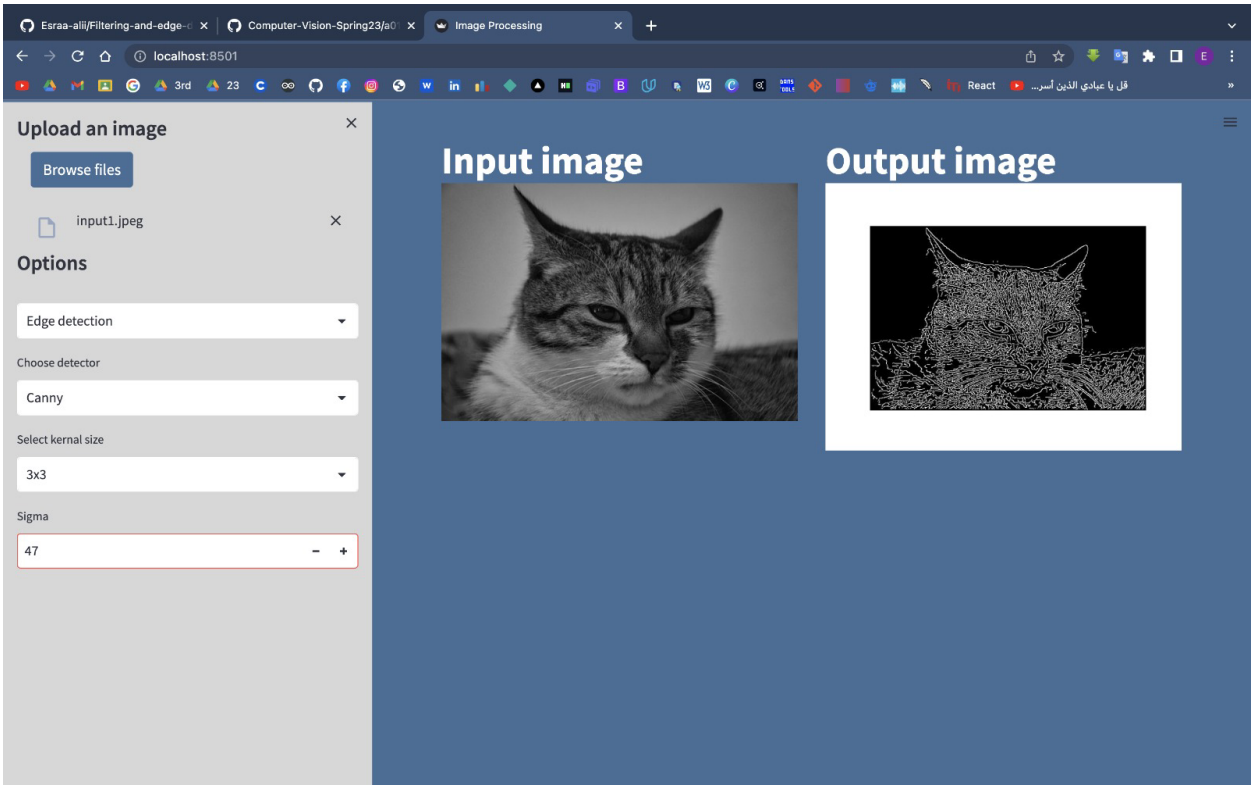
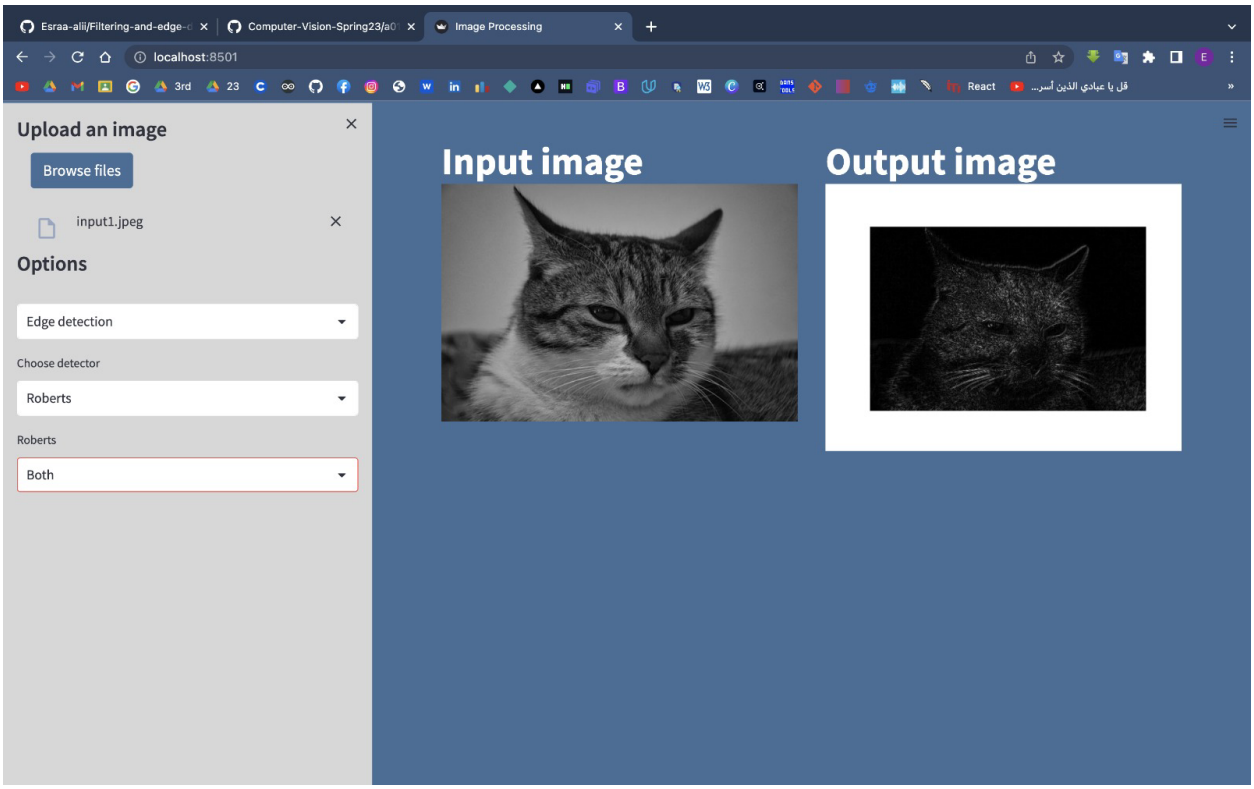
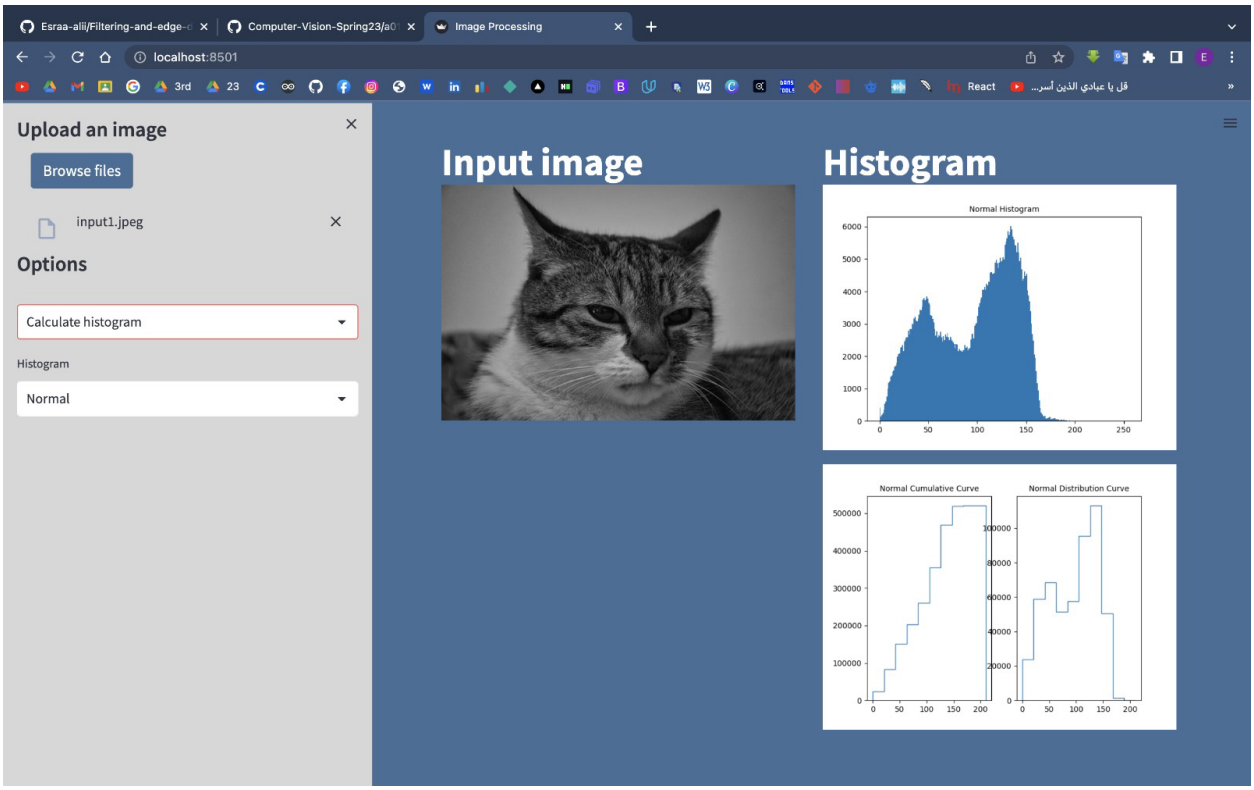


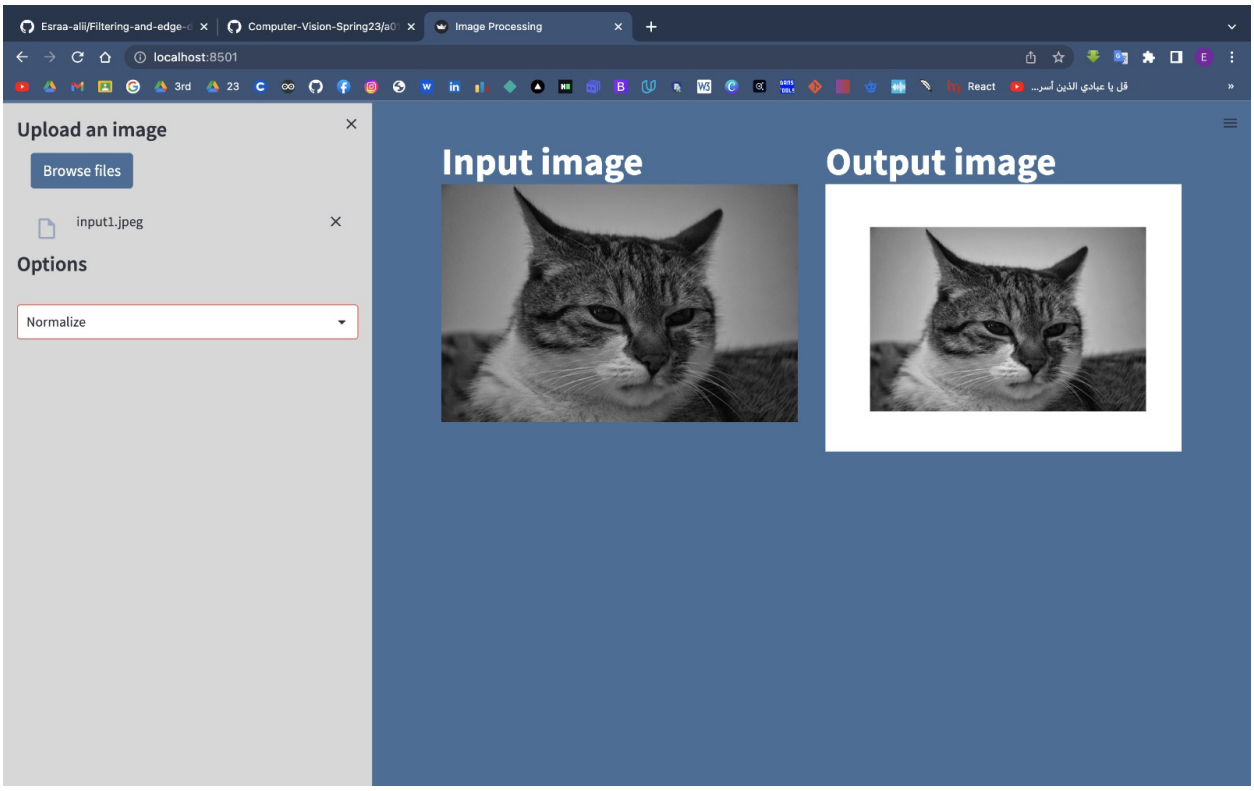
Image Processing
localhost:8501
Upload an image
Browse files
input1.jpeg
Options
Edge detection
Choose detector
Canny
Select kernal size
3x3
Sigma
47
Input image
Output image



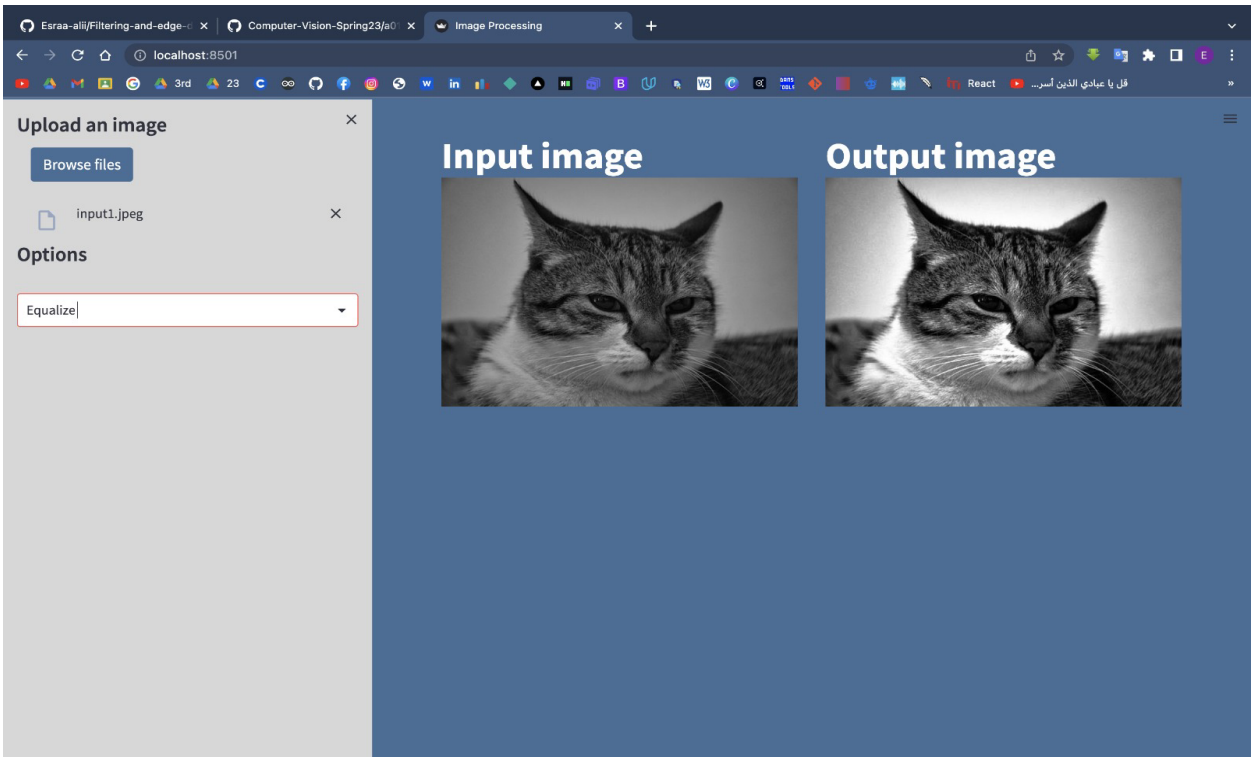
- Draw a histogram and distribution curve for the image.



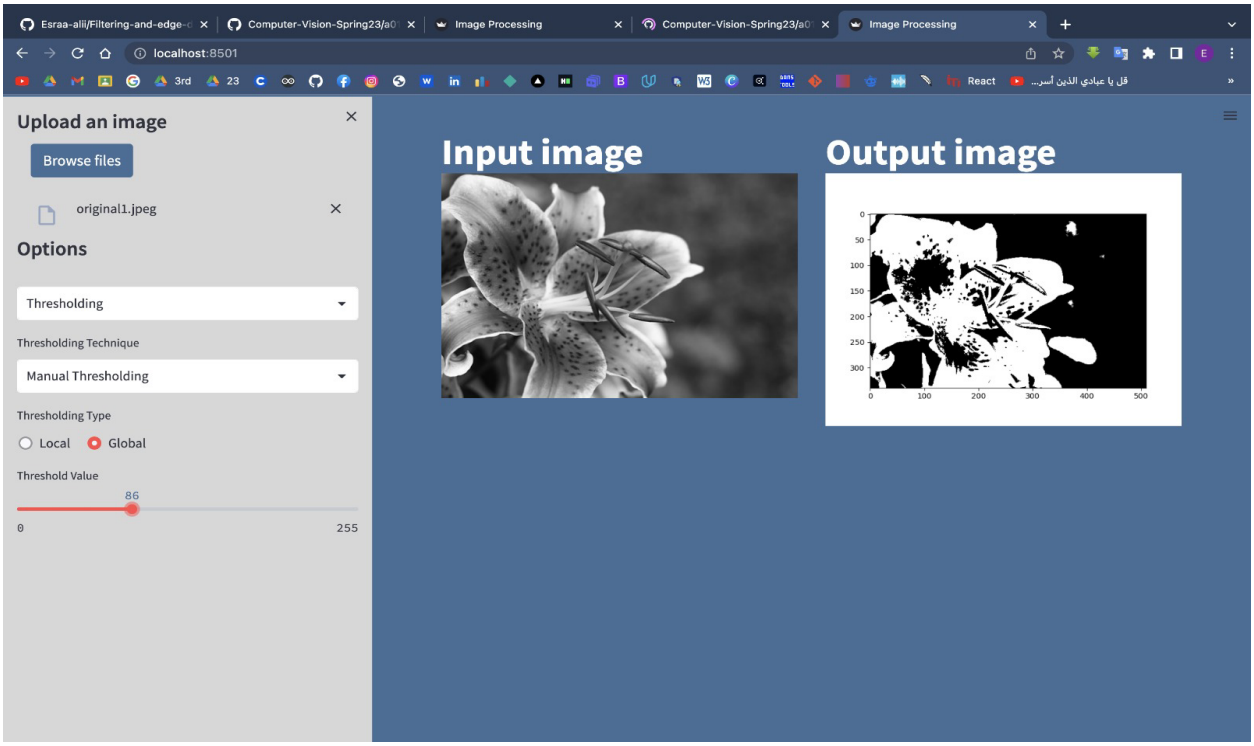
- Normalize the image using the following equation (Output_channel = 255 * (Input_channel - min) / (max-min)).



- Equalize the image as described in algorithm.



- Make the local and global threshold.



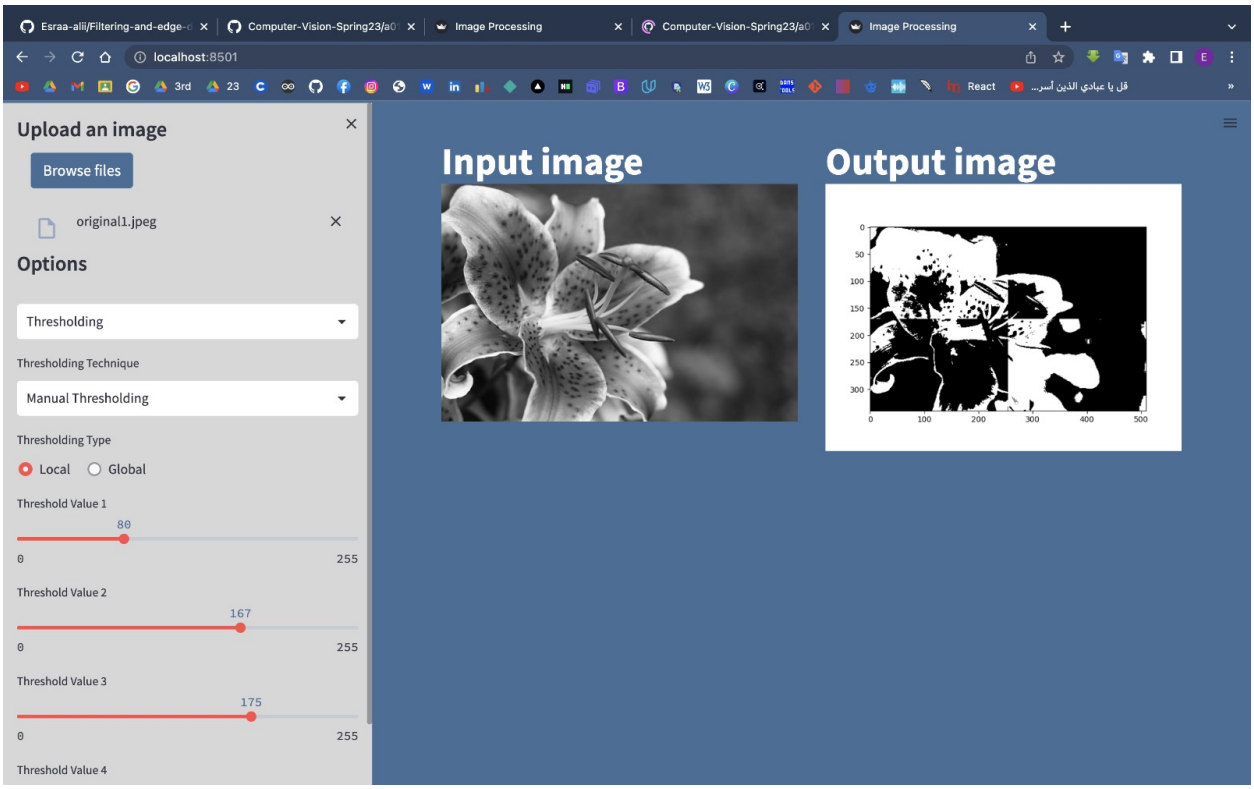


Image Processing
localhost:8501
Upload an image
Browse files
original1.jpeg
Options
Thresholding
Thresholding Technique
Manual Thresholding
Thresholding Type
Local
Global
Threshold Value 1
80
0
255
Threshold Value 2
167
0
255
Threshold Value 3
175
0
255
Threshold Value 4
Input image
Output image

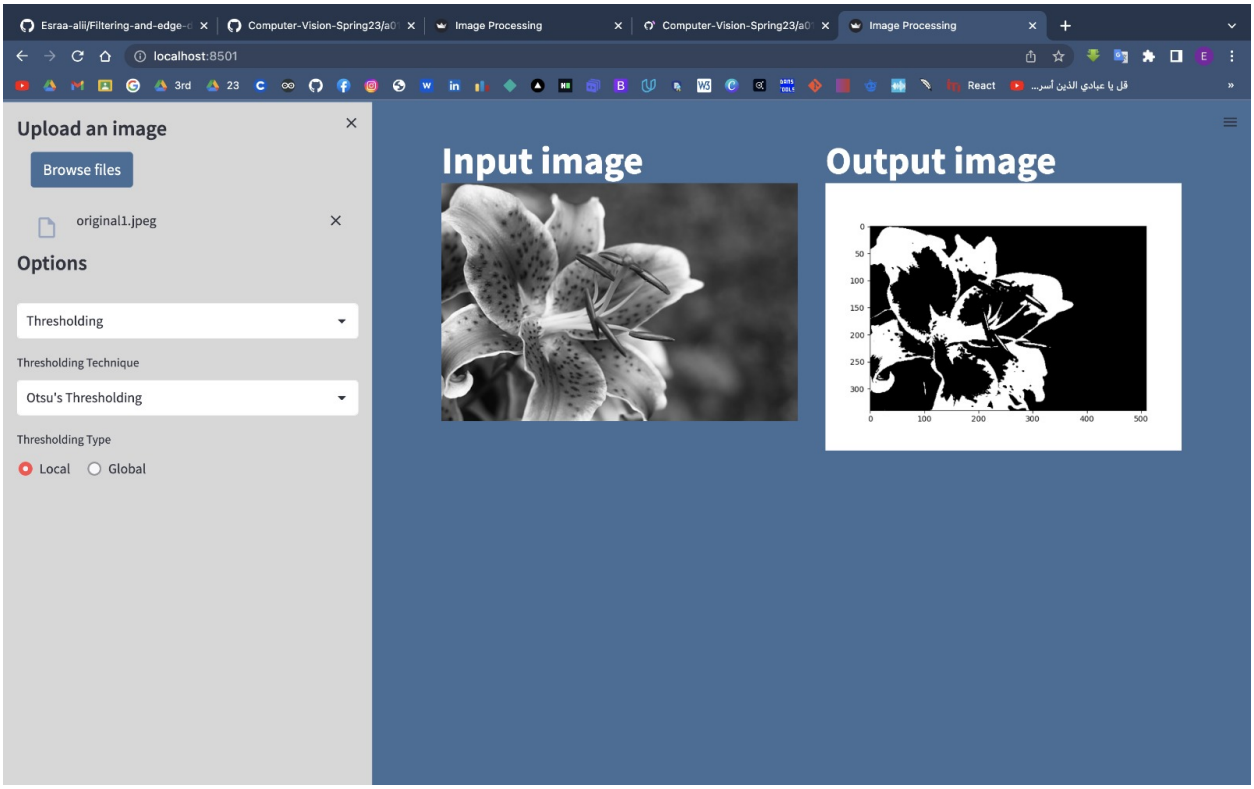
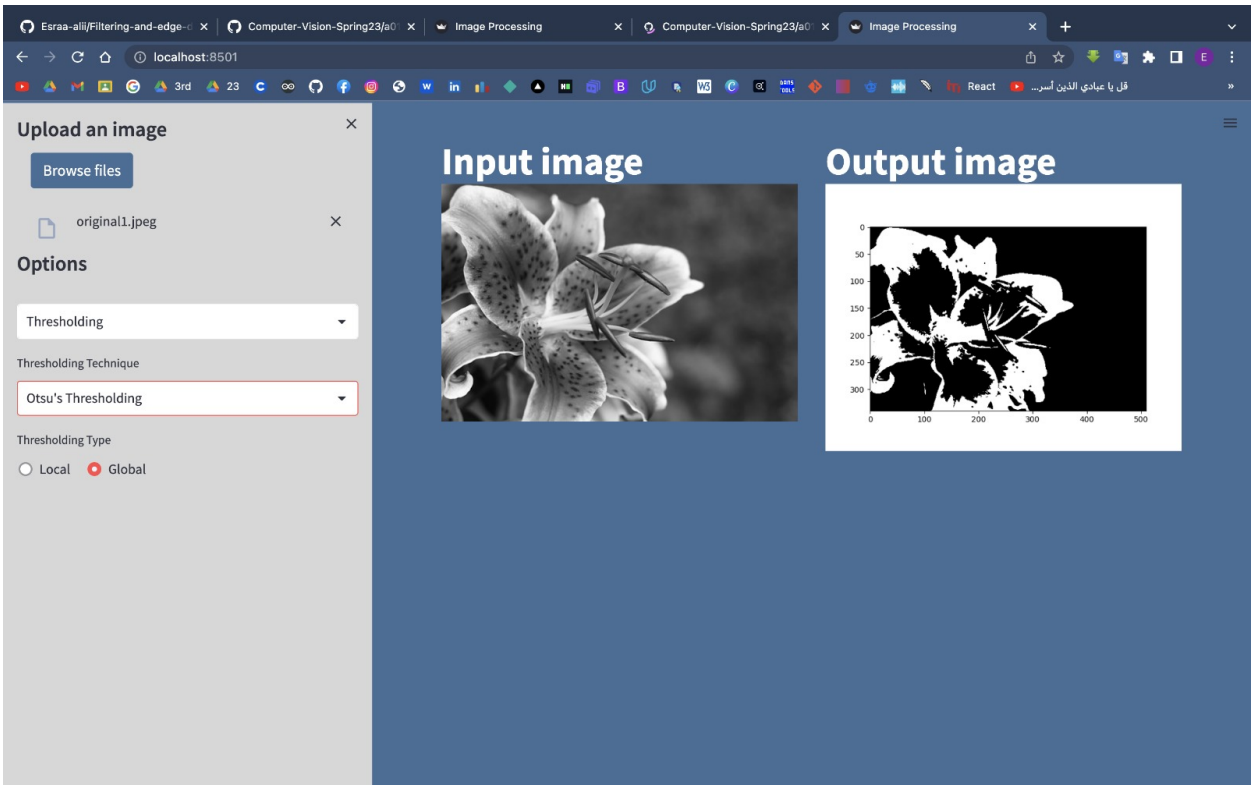
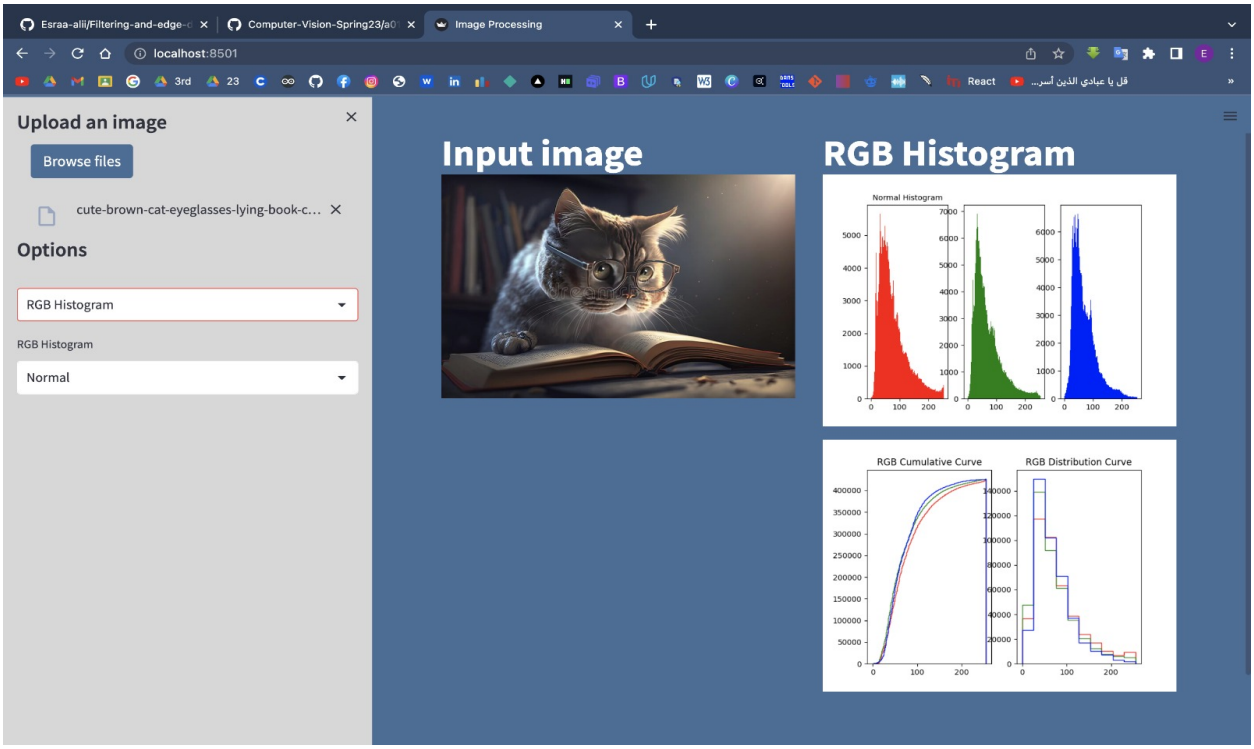
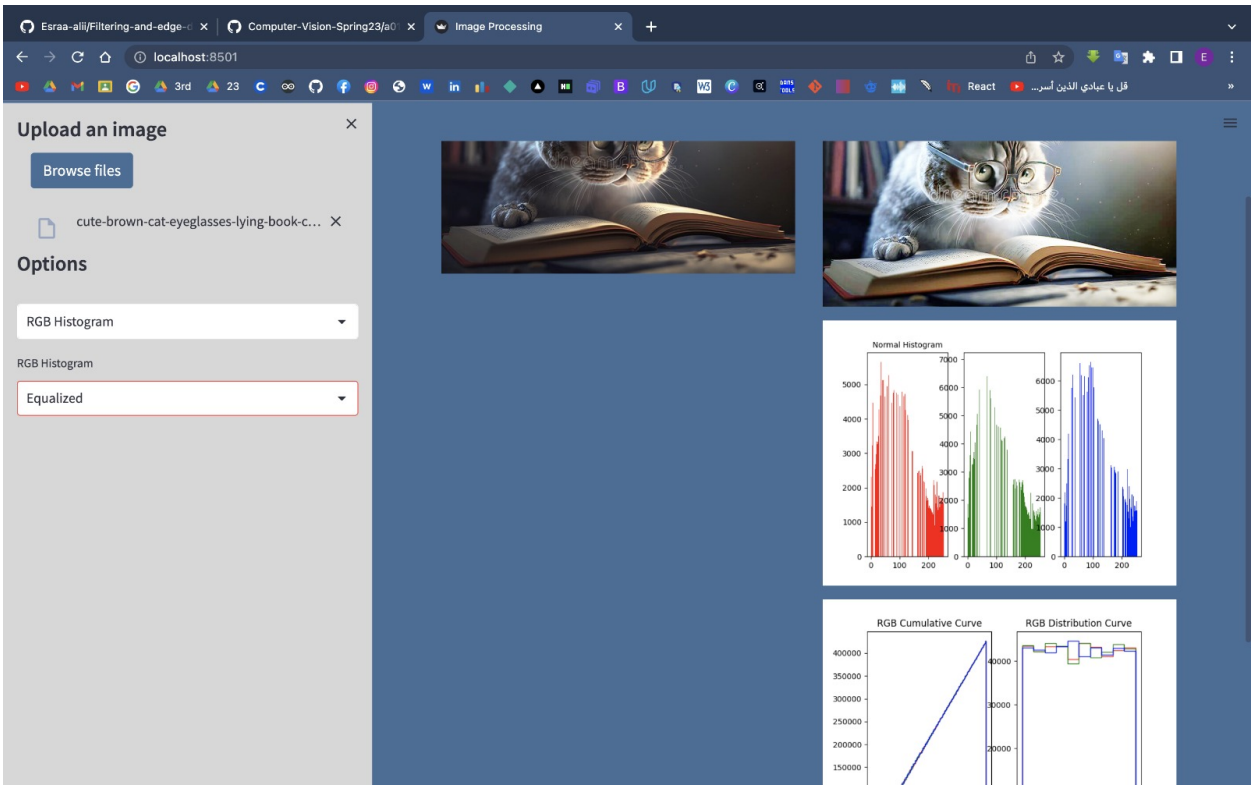


Image Processing
localhost:8501
Upload an image
Browse files
original1.jpeg
Options
Thresholding
Thresholding Technique
Otsu's Thresholding
Thresholding Type
Local
Global
Input image
Output image

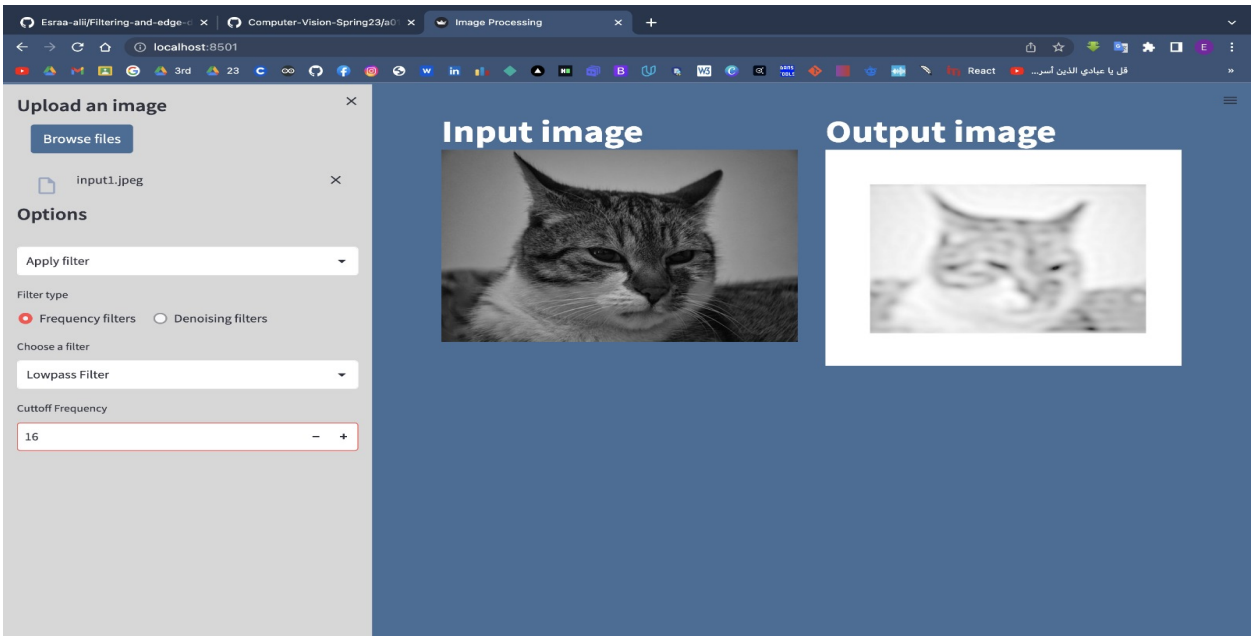


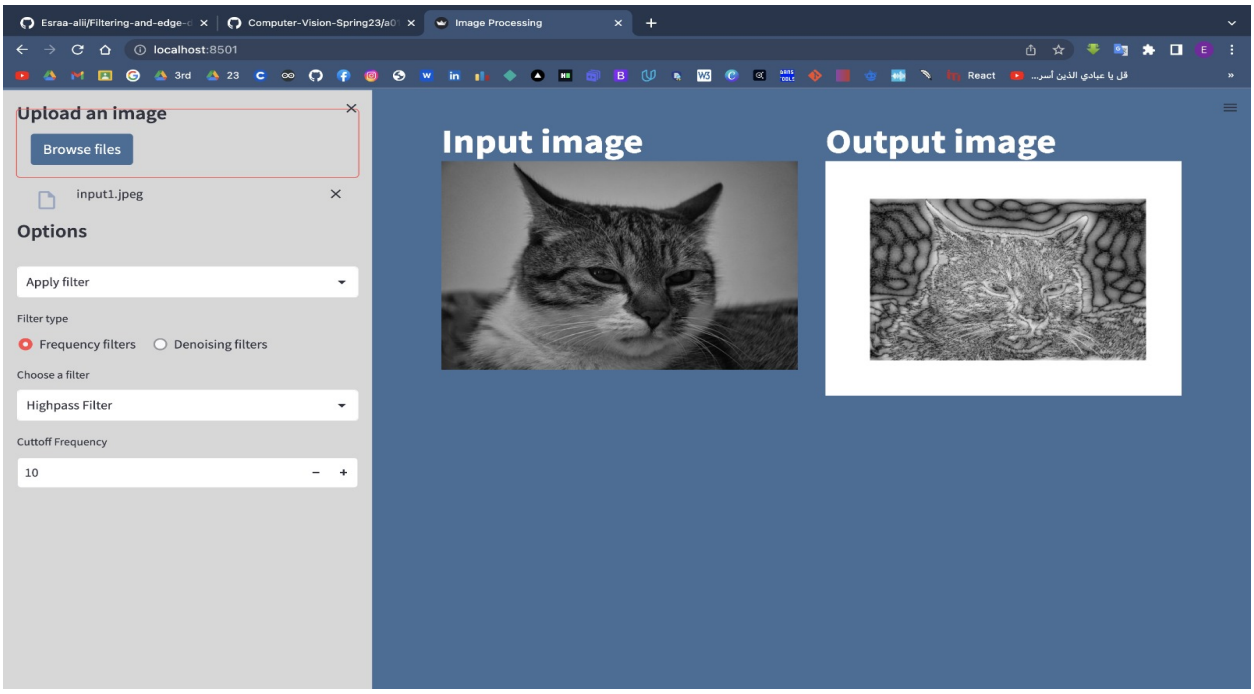
- Transform our image from colored image to gray scale image and plot R, G and B histogram.



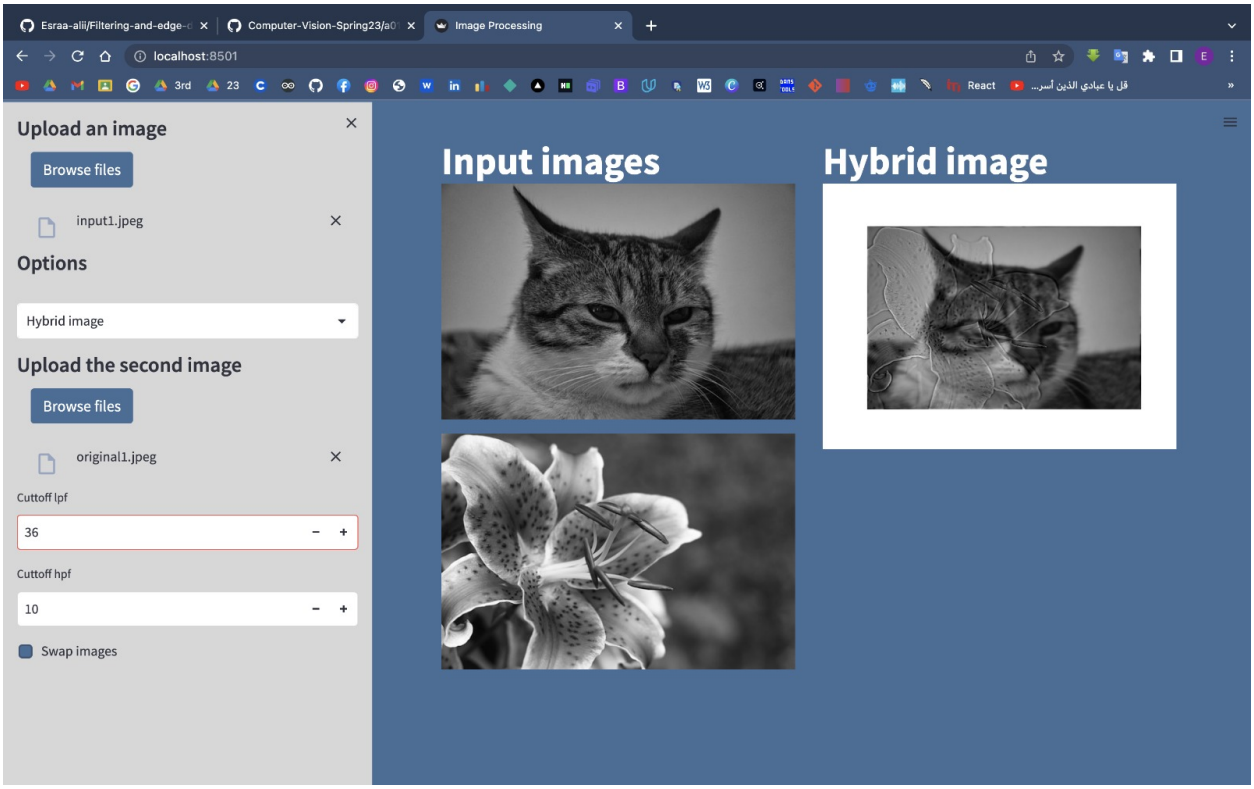


- Apply high and low pass filter to the image.





- Hybrid image.



3. References:

- https://www.sciencedirect.com/science/article/abs/pii/003132039290121X
- https://ieeexplore.ieee.org/abstract/document/663510
- https://ieeexplore.ieee.org/abstract/document/4767769
- https://www.sciencedirect.com/science/article/abs/pii/0165168486900472
- https://ieeexplore.ieee.org/abstract/document/1168349
- https://www.sciencedirect.com/science/article/abs/pii/026288569190025K
- https://sbme-tutorials.github.io/2018/cv/notes/4_week4.html
- http://alumni.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf