

April 18, 2024

Third Year – 2nd Term

Systems & Biomedical Eng. Department

Cairo University

**Team 5**

Salma Ashraf

Sarah Mohamed

Habiba Salama

Hager Samir

Aya Eyad

**computer Vision**

task 2: corner detection and template matching using the sift algorithm

# Harris and λ- Operator

## apply\_harris\_operator:

**Parameters:**

img (np.ndarray): The input image to which the Harris operator will be applied.

k (float): Sensitivity factor to separate corners from edges. Smaller values of k result in the detection of sharper corners.

**Main Point:**

This function applies the Harris corner detection algorithm to the input image. It computes the Harris response for each pixel in the image based on the gradient of the image.

**Observations:**

The function first converts the input image to grayscale.

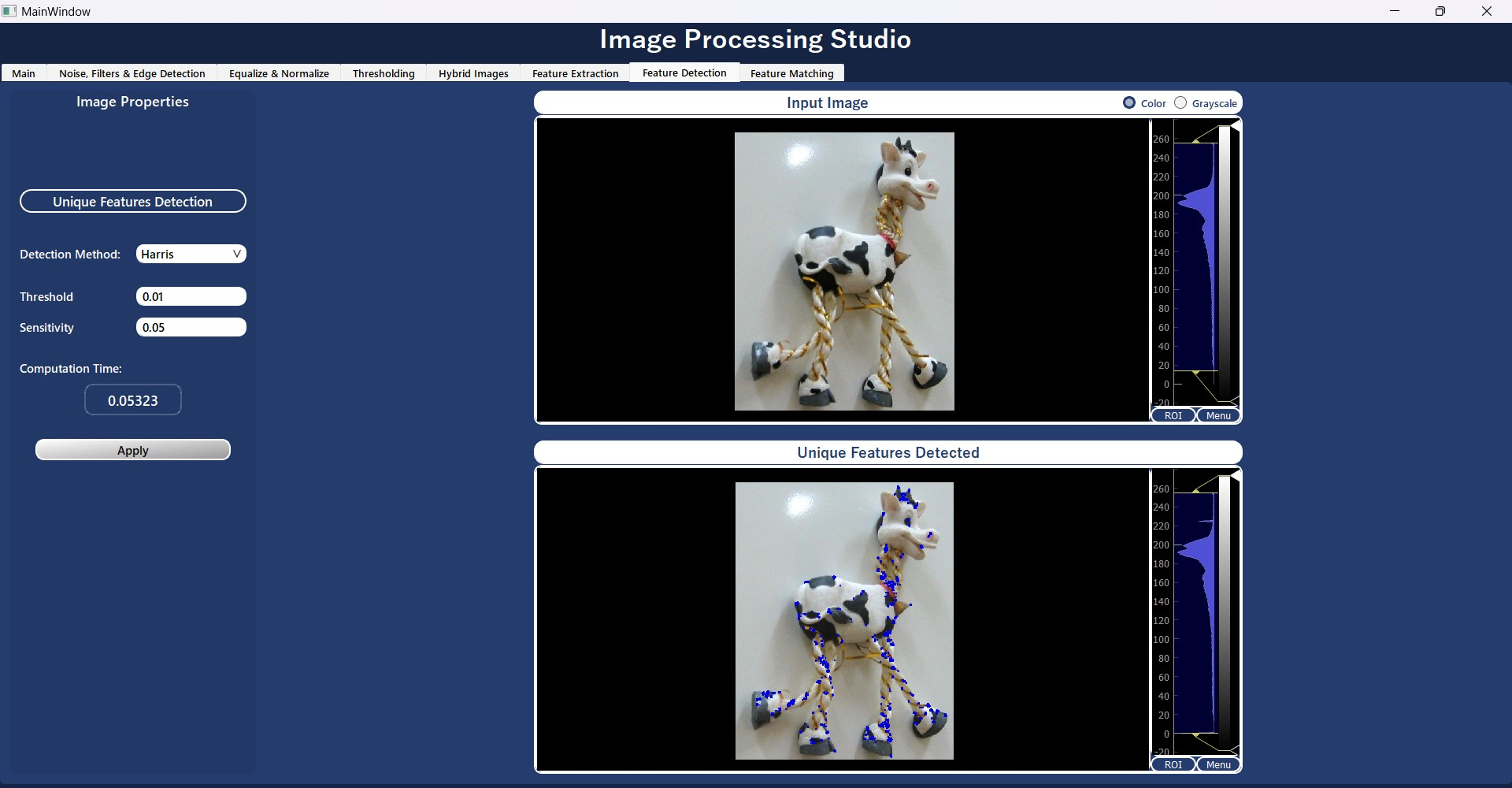
It then computes the gradients of the image using the Sobel operator.

Gaussian blur is applied to the gradients to reduce noise.

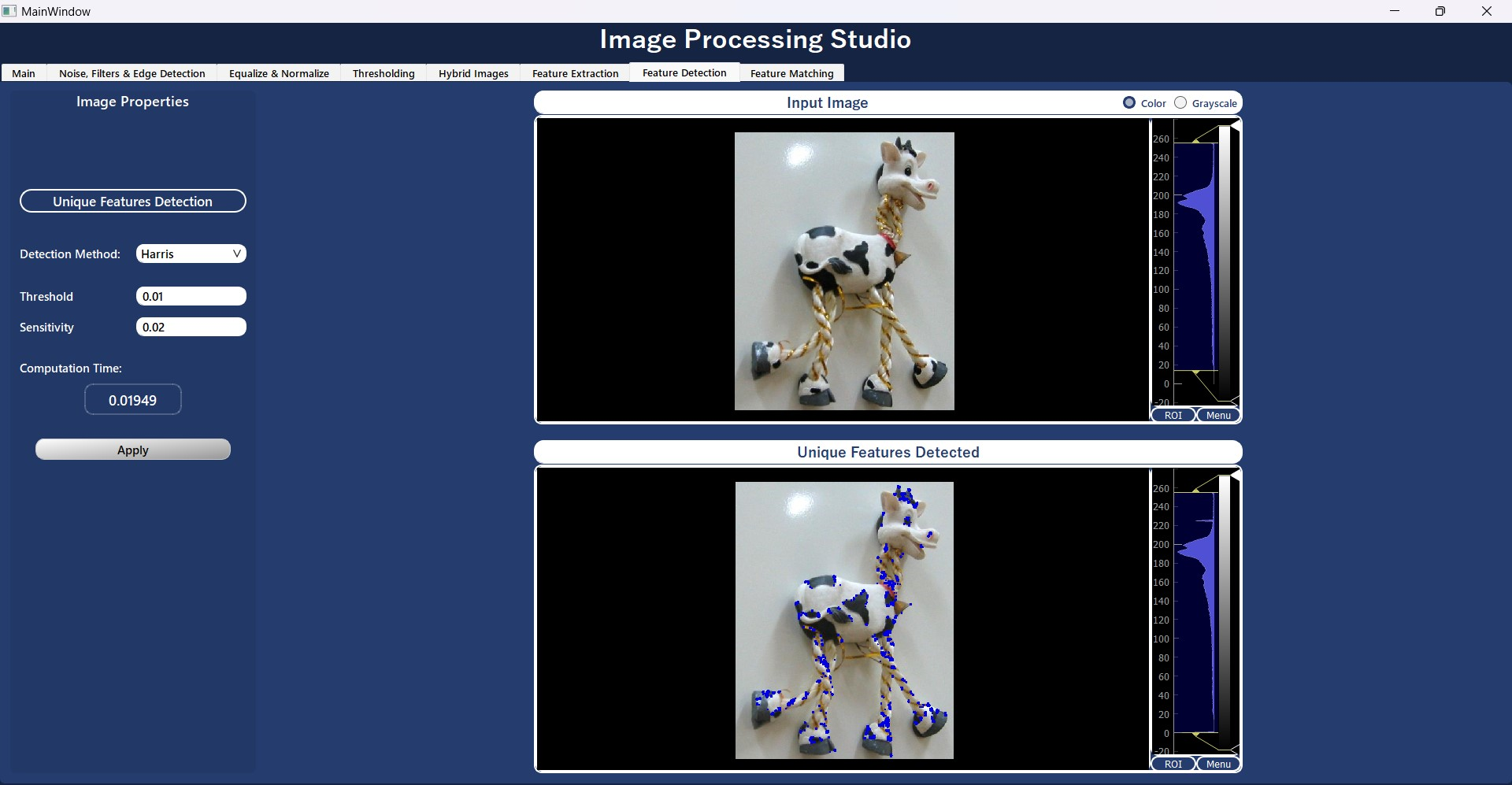
Finally, the Harris response is calculated using the determinant and trace of the gradient matrix.

**Experiment:**

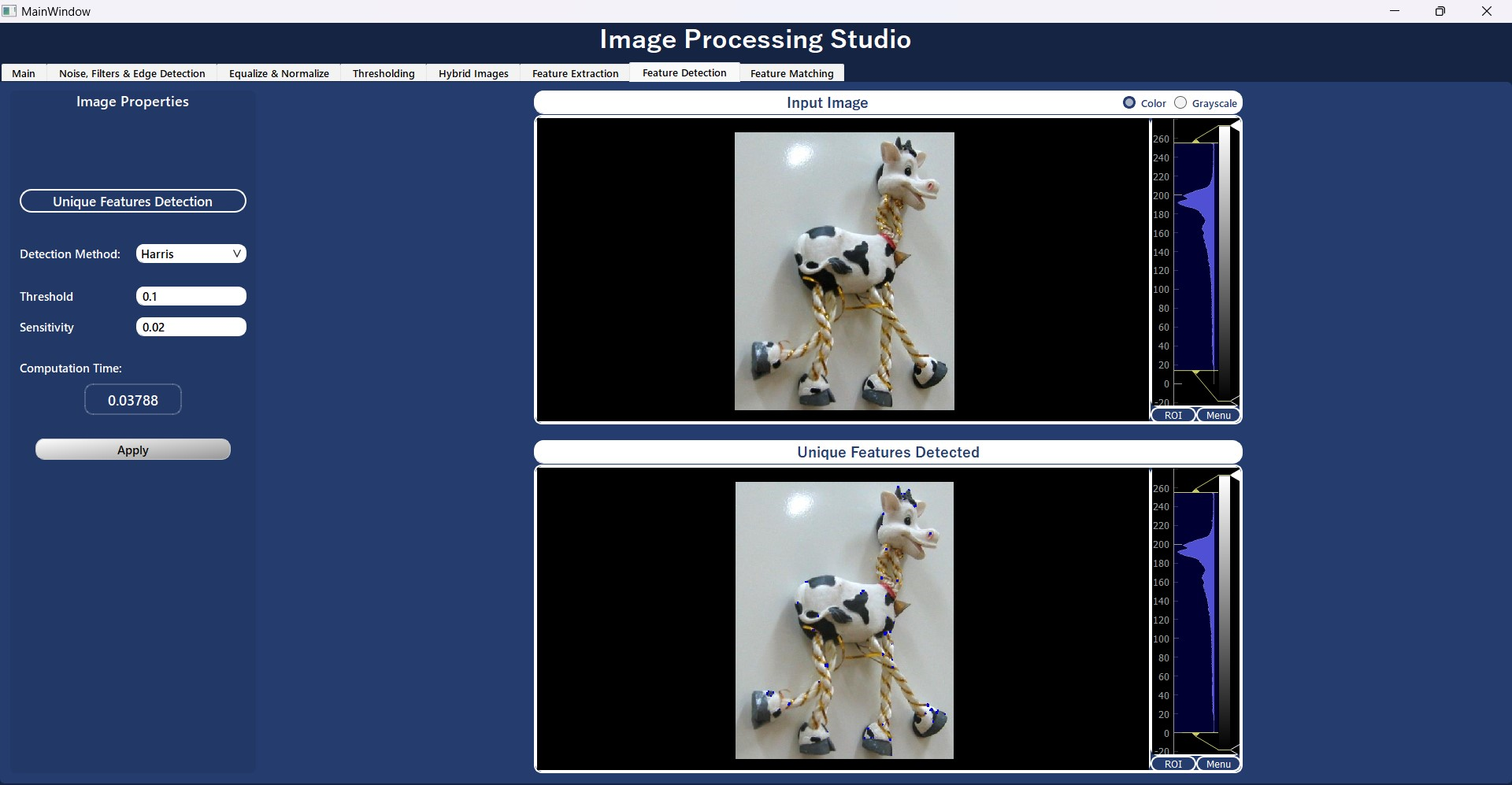
* **Apply Harris operator with TH= 0.01 & and K= 0.05 :**



* **Apply Harris operator with TH= 0.01 & and K= 0.02 (Small value of K detect sharp corners) :**



* **Apply Harris operator with TH= 0.1 & and K= 0.02 (large value of TH detect less corners) :**



## apply\_lamda\_minus\_operator:

**Parameters:**

img (np.ndarray): The input image to which the λ- operator will be applied.

**Main Point:**

This function applies the λ- corner detection operator to the input image. It computes the λ- response for each pixel in the image based on the gradients of the image.

**Observations:**

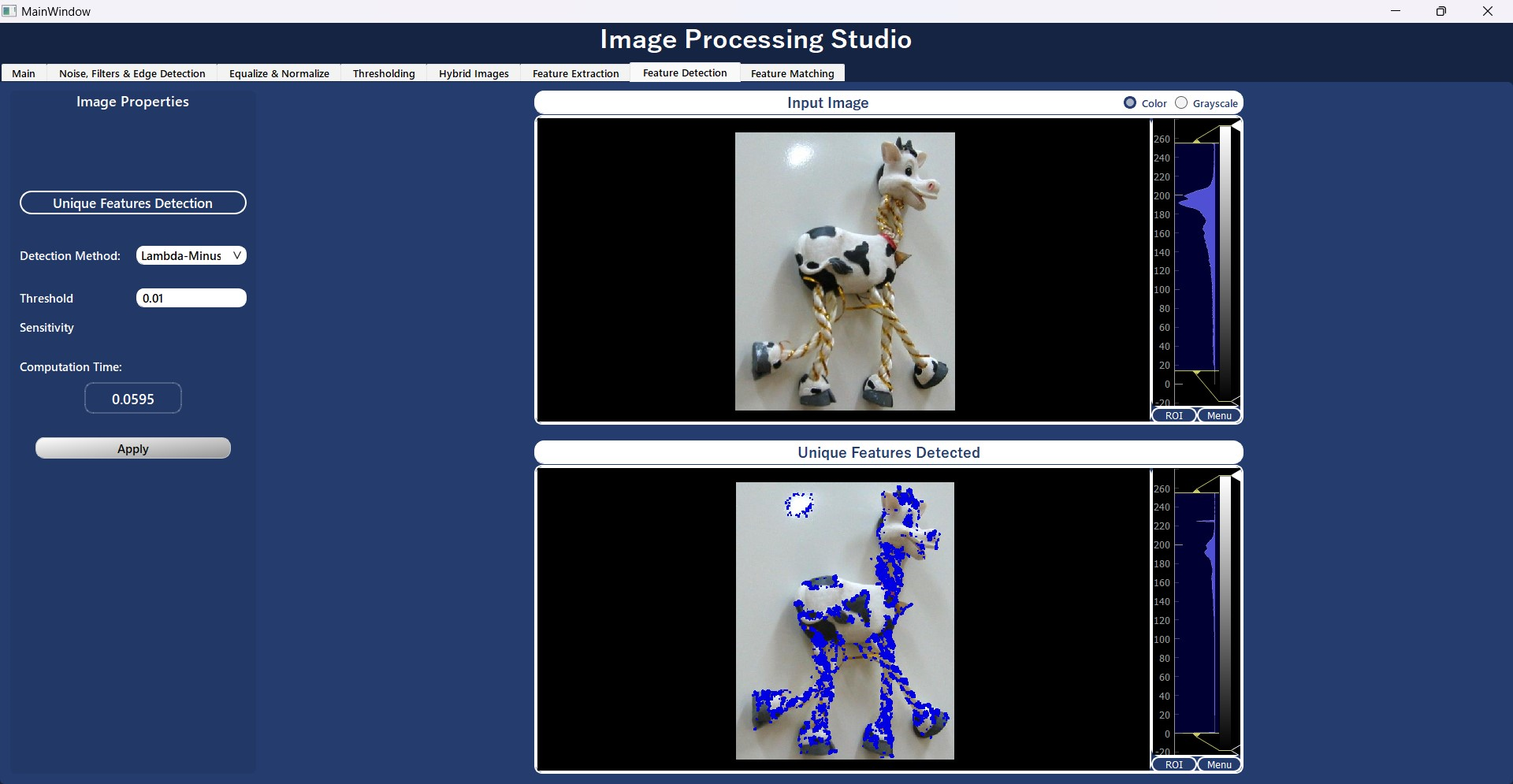
Similar to the Harris operator, it starts by converting the input image to grayscale and computing gradients.

Instead of directly calculating the Harris response, it computes the sum of squared gradients within a window around each pixel to find the corner response.

Eigenvalues of the gradient matrix are calculated to obtain the λ- response.

**Experiment:**

* **Apply Lamda Minus operator with TH= 0.01 (take more time than harris) :**



## get\_operator\_indices:

**Parameters:**

operator\_response (np.ndarray): An array representing the Operator response of an image.

threshold (float, optional): Threshold value used for computing local maxima.

**Main Point:**

This function identifies corners, edges, and flat areas in the image based on the Operator response.

**Observations:**

It dilates the Operator response to make the points clearer.

Corner indices are identified where the Operator response is greater than a certain threshold.

Edge indices are identified where the Operator response is negative.

Flat indices are identified where the Operator response is zero.

## map\_indices\_to\_image:

**Parameters:**

img (np.ndarray): The input image on which dots will be drawn.

indices (np.ndarray): An array of 0's and 1's representing indices of interest.

color (list): Color to draw the dots with, specified as a list

**Main Point:**

This function draws dots on the input image based on provided indices.

**Observations:**

It creates a copy of the input image to avoid modifying the original.

Dots are drawn on the image where the indices are equal to 1, using the specified color.

The function returns a new image with dots drawn based on the provided indices.

# SIFT

**The SIFT algorithm can be decomposed into four steps:**

1. Scale space formation
2. Keypoint detection.
3. Orientation assignment.
4. Feature descriptor generation.
5. **Scale space formation**

Time taken for this step: 0.4591035842895508 seconds

## construct\_scale\_space:

**Parameters:**

image (np.ndarray): The input image to construct the different scales and octaves.

sigma\_values: The standard deviation of gaussian kernels forming the scales and it equals (k^i)\*sigma, where sigma = 1.6 and k = sqrt(2). These numbers were rcommendedby the paper.

**Returns:**

diff\_of\_gaussians (2d list): A list of lists carrying the DOGs of all octaves (4x4).

octaves (list): The list of 4 octaves of different resolutions.

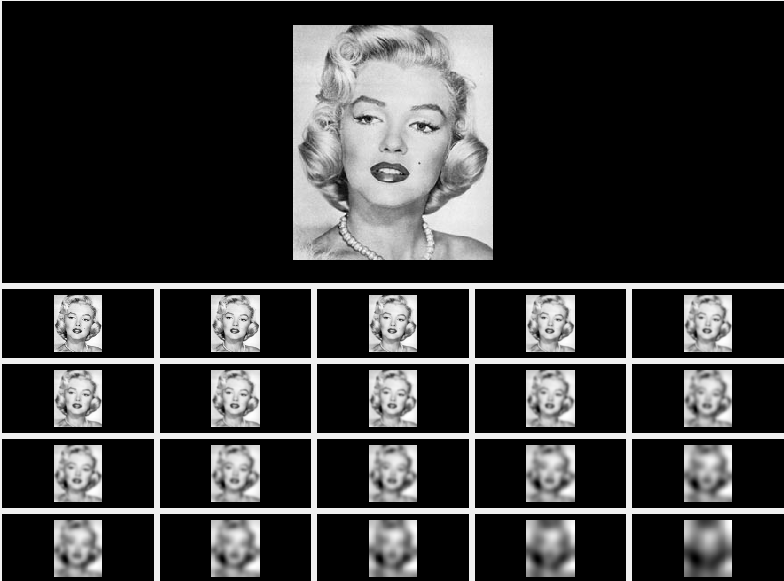
**Steps:**

Apply gaussian kernel with the list of sigma values recommended by the paper to get the octave’s starting image.

Subtract every two successive scales to get the DOGs.

For the upcoming octave, decrease the size (resolution) by half.

**Experiment:**

* **Constructing 4 octaves, each of 5 scales:** ****

## Key point detection

## compare\_with\_neighbors:

**Parameters:**

stage\_list (np.ndarray): A list of stages for one octave.

**Returns:**

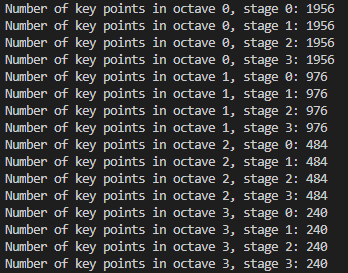
stage\_list (np.ndarray): The list after modification (extraction of key points)

**Steps:**

For every point in the image not equal to zero (key point candidate), compare it to the points in the 3x3 kernel above and below it and in the same stage, unless it is their maximum or minimum value, we zero it and it is discarded from being a key point, points in the upper and lowermost octaves are compared to points in the same stage and the stage above/below it. Then points are thresholded at 0.03 (recommended by the paper) to eliminate points of slight differences between scales.

**Experiment:**

The number of key points produced for every octave and stage

****

# Descriptors Calculation

**Orientation\_assignment:**

**Parameters:**

keypoints\_list (list): A list of tuples containing information about keypoints, each tuple consists of (octave\_index, sigma\_value, keypoint\_coordinates).

list\_octaves (list): List of Gaussian octaves generated from the input image.

**Description:**

Assigns orientation to key points based on gradient information.

**Steps:**

1. Iterate over each octave index and each keypoint in the keypoints\_list.

4. Retrieve the octave and scale information for the keypoint.

5. Calculate the Gaussian kernel parameters based on the scale.

6. Compute the gradient magnitude and orientation for the keypoint's scale.

7. Iterate over each keypoint coordinate.

8. Compute the orientation bin index for the gradient orientation.

9. Define the window around the keypoint for gradient calculation.

10. Extract the gradient magnitude and orientation information within the window.

11. Weight the gradient magnitude with the Gaussian kernel.

12. Compute a histogram of weighted magnitudes based on orientation bins.

13. Determine the dominant orientation as the bin with the maximum magnitude.

14. Append the keypoint and its dominant orientation to the dominant\_dir\_list.

# Feature Matching

**calculate\_SSD:**

**Parameters:**

descriptor1: The first of the 2 descriptors to calculate their error using SSD.

descriptor2: The second of the 2 descriptors to calculate their error using SSD

**Returns:**

ssd\_value (float): The error of the 2 descriptors.

**calculate\_NCC:**

**Parameters:**

descriptor1: The first of the 2 descriptors to calculate their error using normalized cross correlation.

descriptor2: The second of the 2 descriptors to calculate their error using normalized cross correlation.

**Returns:**

ncc\_value (float): The error of the 2 descriptors.

**Observation:**

SSD is faster and has less complexity than the NCC, although NCC is more robust to intensity variations.

**Find\_best\_match:**

Finds the best match for a given SIFT descriptor among a set of descriptors.

**Parameters:**

descriptor (list): A single SIFT descriptor.

descriptors (list): List of SIFT descriptors to search for matches.

**Returns:**

int or None: The index of the best matching descriptor in descriptors,

or None if no match is found.

**Parameters:**

descriptor1: The first of the 2 descriptors to calculate their error using SSD.

descriptor2: The second of the 2 descriptors to calculate their error using SSD

**Returns:**

ssd\_value (float): The error of the 2 descriptors.

**Match\_descriptors:**

**Parameters:**

descriptors1 (list): List of SIFT descriptors from the first image.

descriptors2 (list): List of SIFT descriptors from the second image.

**Returns:**

matches (list): A list of tuples containing indices of matched descriptors. Each tuple consists of the index of the descriptor in descriptors1 and the index of its best match in descriptors2.