



# Computer Vision

## Task 3

### Team 18

Ahmed Kamal Mohamed	Sec.1
Amgad Atef Abd Al-Hakeem	Sec.1
Mahmoud Mohamed Ali	Sec.2
Mahmoud Magdy Mohamed	Sec.2
Mohanad Emad El-Sayed	Sec.2

**Supervised By**  
Dr. Ahmed Badawi  
Eng. Lila Abbas  
Eng. Omar Hesham

# TABLE OF CONTENTS

<b>1. Extract Features:</b>	<b>3</b>
<b>1.1 Harris operator:</b>	<b>3</b>
1.1.1 Principle:	3
1.1.2 Pseudocode:	3
1.1.3 Output image:	4
<b>1.2 lambda (<math>\lambda</math>-) operator:</b>	<b>4</b>
1.2.1 Principle:	4
1.2.2 Pseudocode:	5
1.2.3 Output image:	5
<b>2. Generate Features:</b>	<b>6</b>
<b>2.1 Key points generator with Scale Invariant Feature (SIFT):</b>	<b>6</b>
2.1.1 Principle:	7
2.1.2 Pseudocode:	7
2.1.3 Output image:	7
<b>3. Image Matching:</b>	<b>8</b>
<b>3.1 Image matching using Scale Invariant Feature (SIFT):</b>	<b>8</b>
3.1.1 Pseudocode:	8
3.1.2 Output image:	9
3.1.3 Time:	10
<b>3.2 Template matching using similarity:</b>	<b>10</b>
3.2.1 Principle:	11
3.2.2 Pseudocode:	11
3.2.3 Output image:	11
3.2.4 SSD time Vs NCC time:	13

# 1. Extract Features:

## 1.1 Harris operator:

The Harris corner detector is a technique in computer vision used for identifying corner points in images.

### 1.1.1 Principle:

The Harris corner detector works by analyzing the variations in intensity in different directions around each pixel. It computes a score based on the amount of change in intensity when the image is shifted by a small amount in any direction. Points with high scores indicate corners.

### 1.1.2 Pseudocode:

1. Change the input image to gray scale.
2. Apply gaussian filter to blur the image.
3. Compute the gradients  $I_x$  and  $I_y$  using a gradient operator (Schar operator).
4. Compute matrices  $I_x^2$ ,  $I_y^2$  and  $I_x * I_y$ .
5. Compute this matrix for each pixel:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Then compute Harris response from equation:

$$R = \det(M) - k(\text{trace}(M))^2$$

6. Find the corner point which value is bigger than the neighbors pixels values and bigger than a threshold.

### 1.1.3 Output image:

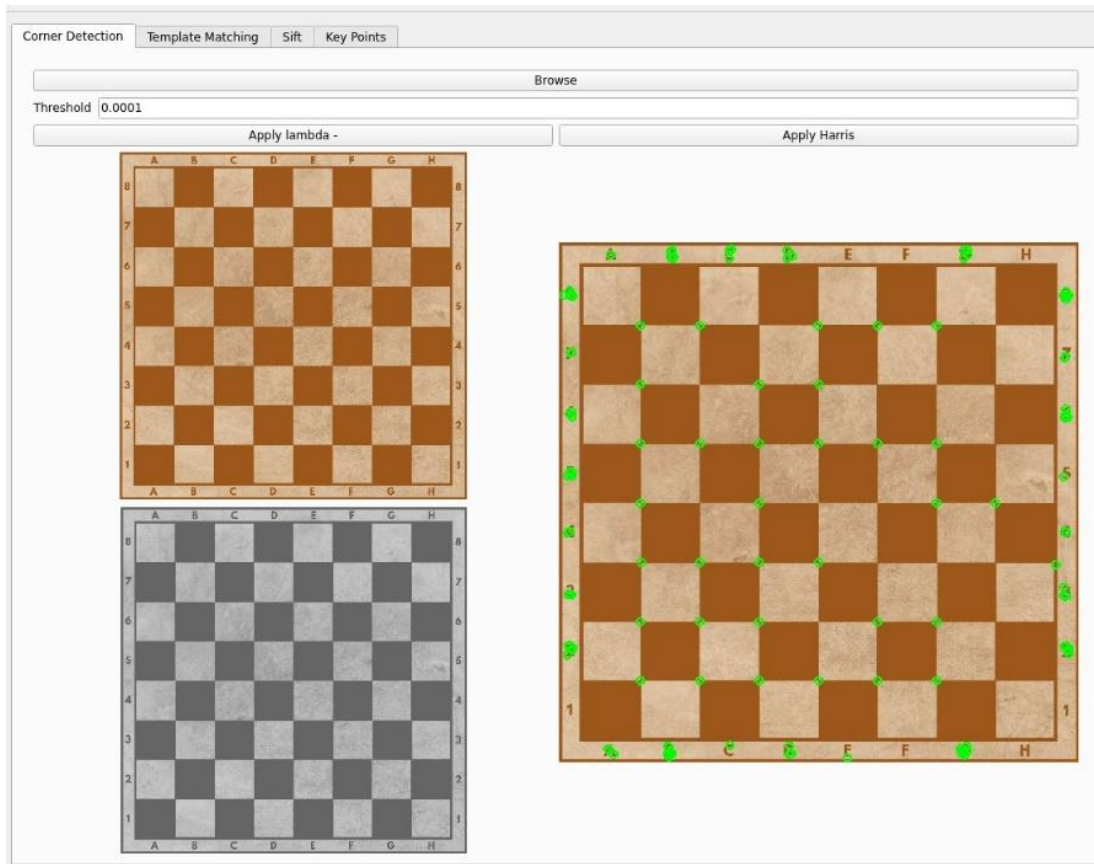


Figure 1: Harris operator.

## 1.2 lambda ( $\lambda$ -) operator:

Lambda minus is an operator used for detecting dark structures in an image. It's particularly useful for detecting thin and elongated structures such as blood vessels, cracks, or edges.

### 1.2.1 Principle:

The  $\lambda$ - operator computes the local minimum of the image with a structuring element (often a line or a thin rectangle). It highlights the dark regions that fit the shape of the structuring element.

### 1.2.2 Pseudocode:

1. Change the input image to gray scale.
2. Apply gaussian filter to blur the image.
3. Compute the gradients  $I_x$  and  $I_y$  using a gradient operator (Scharr operator).
4. Compute matrices  $I_x^2$ ,  $I_y^2$  and  $I_x I_y$ .
5. Compute this matrix for each pixel:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Then compute lambda minus and lambda plus for the image.

6. Find the corner point which value is bigger than the neighbors pixels values and bigger than a threshold.

### 1.2.3 Output image:

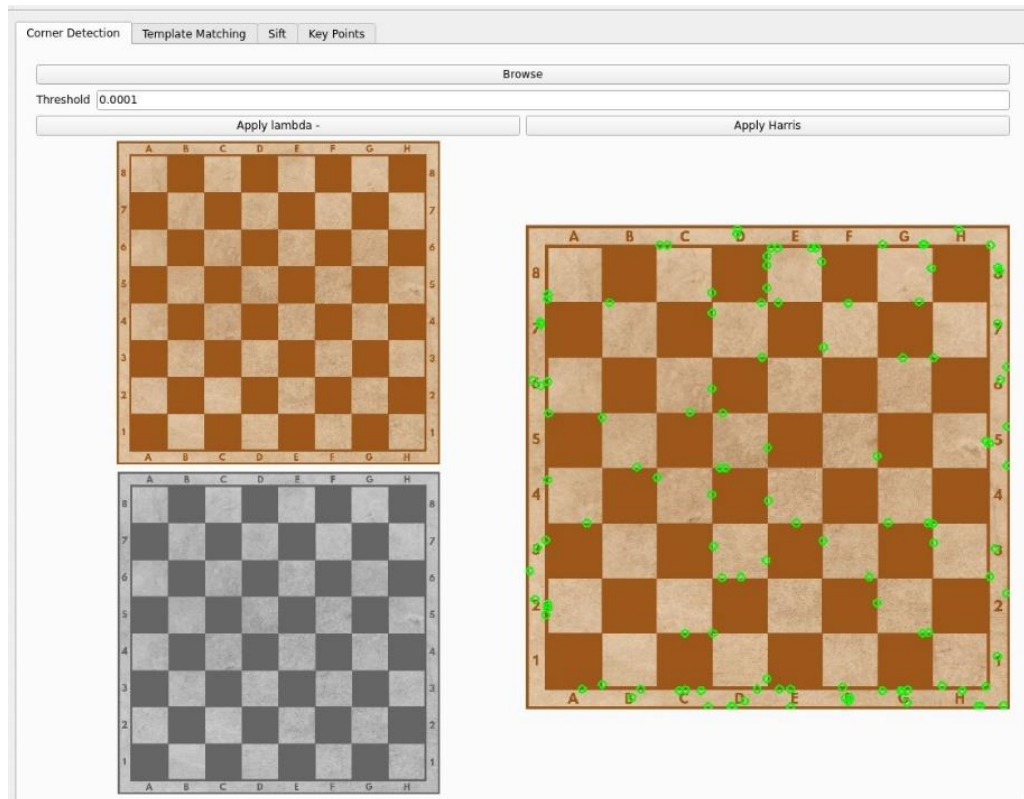


Figure 2: lambda ( $\lambda$ -) operator.

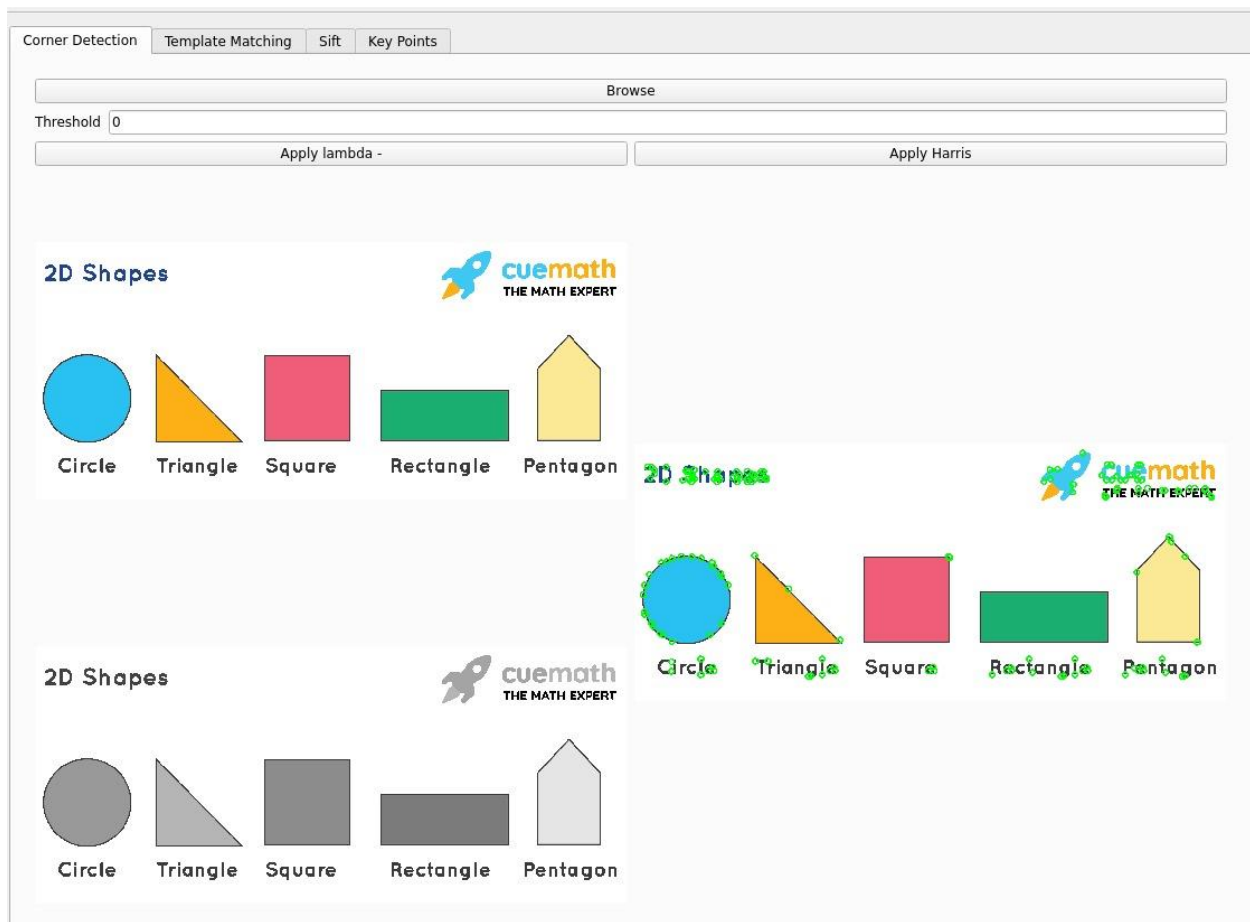


Figure 3: lambda ( $\lambda$ -) operator.

## 2. Generate Features:

### 2.1 Key points generator with Scale Invariant Feature (SIFT):

The Scale-Invariant Feature Transform (SIFT) algorithm is a method used for detecting and describing local features in images. These features, often referred to as key points or interest points, are distinctive regions of an image that can be reliably identified across different scales, rotations, illuminations, and viewpoints.

### 2.1.1 Principle:

SIFT identifies key points that are distinctive across an image's width, height, and most importantly, scale. By considering scale, we can identify key points that will remain stable (to an extent) even when the template of interest changes size, when the image quality becomes better or worse, or when the template undergoes changes in viewpoint or aspect ratio. Moreover, each key point has an associated orientation that makes SIFT features invariant to template rotations. Finally, SIFT will generate a descriptor for each key point, a 128-length vector that allows key points to be compared. These descriptors are nothing more than a histogram of gradients computed within the key point's neighborhood.

### 2.1.2 Pseudocode:

1. Apply gaussian filter to blur the image.
2. Build image pyramid of resized image with scale factor for the image.
3. Detect the key points of image pyramid.
4. Filter the key points with threshold.
5. Compute descriptors for the key points by applying some steps:
  - Compute gradients using Sobel operators.
  - Compute magnitude and orientation of gradients.
  - Iterate over key points and Extract patch around each key point and compute histogram of gradients.
  - Normalize histogram.
  - save normalized histogram as descriptor.

### 2.1.3 Output image:



Figure 4: Key points with SIFT.

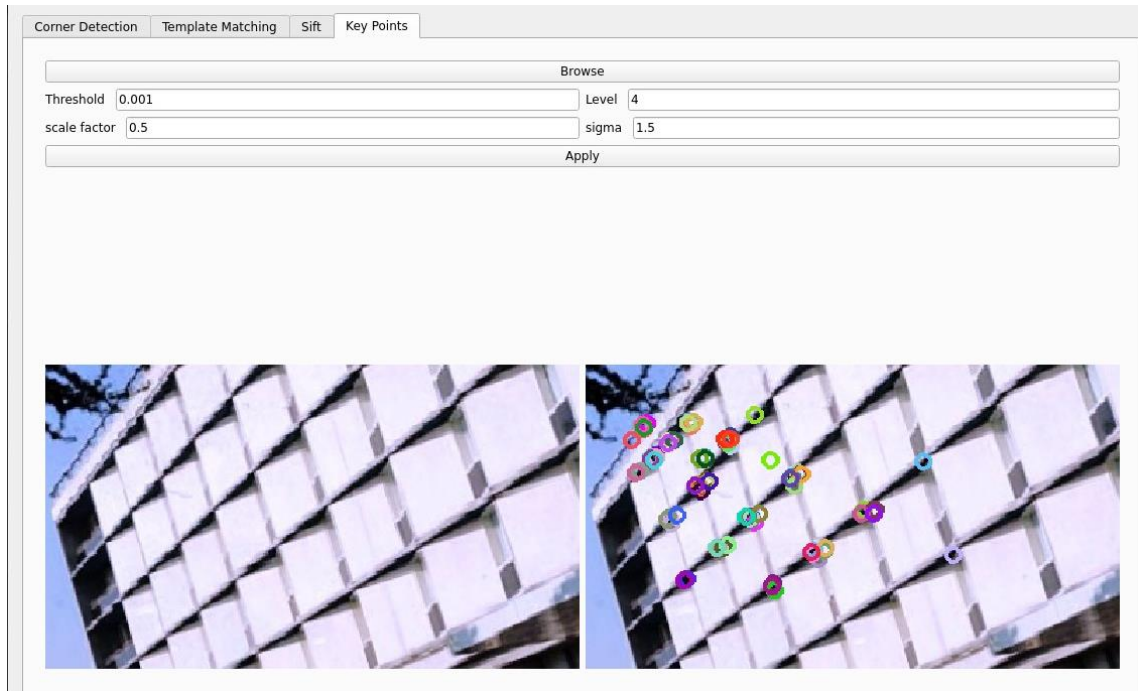


Figure 5: Key points with SIFT.

## 3. Image Matching:

### 3.1 Image matching using Scale Invariant Feature (SIFT):

Using SIFT to match two images with difference scale, illumination, and size.

#### 3.1.1 Pseudocode:

1. Get the key points of each image using SIFT ([2.1](#)).
2. Match the descriptors of the two images either by Sum of Square Distance (SSD) or Normalized Cross Correlations (NCC)
3. Draw the matching image.



### 3.1.2 Output image:

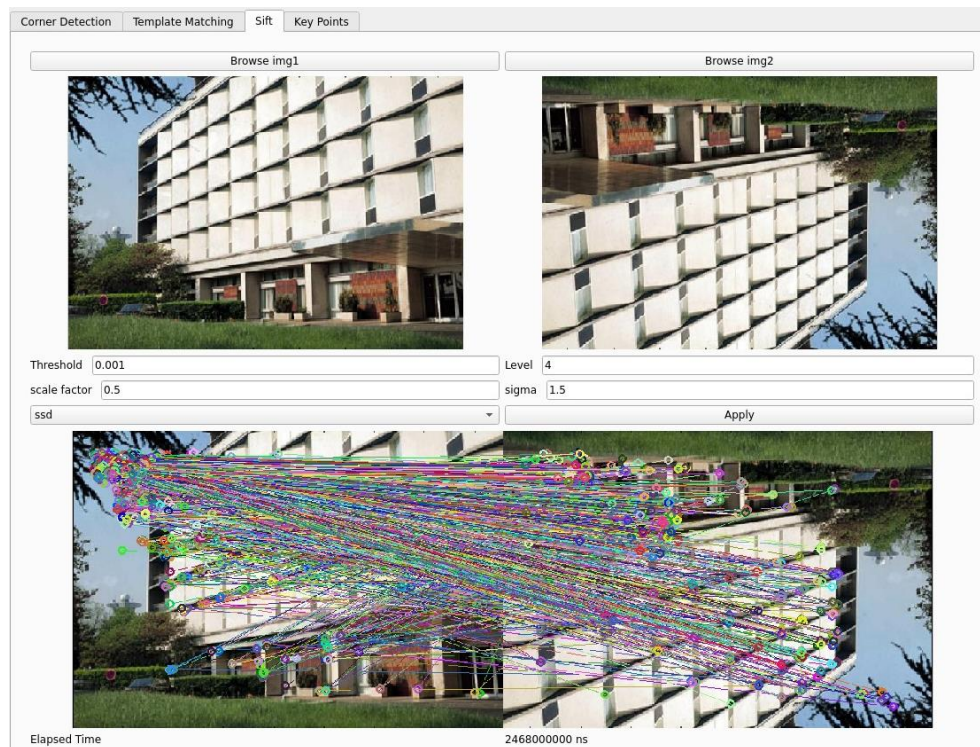


Figure 6: Image matching using SIFT (SSD).

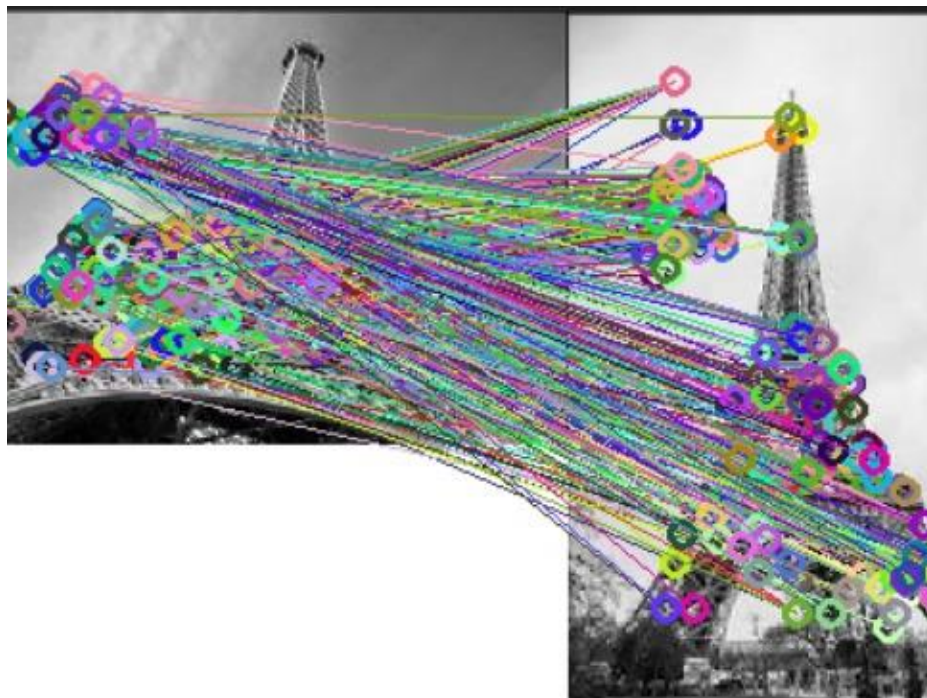


Figure 7: Image matching using SIFT(NCC).

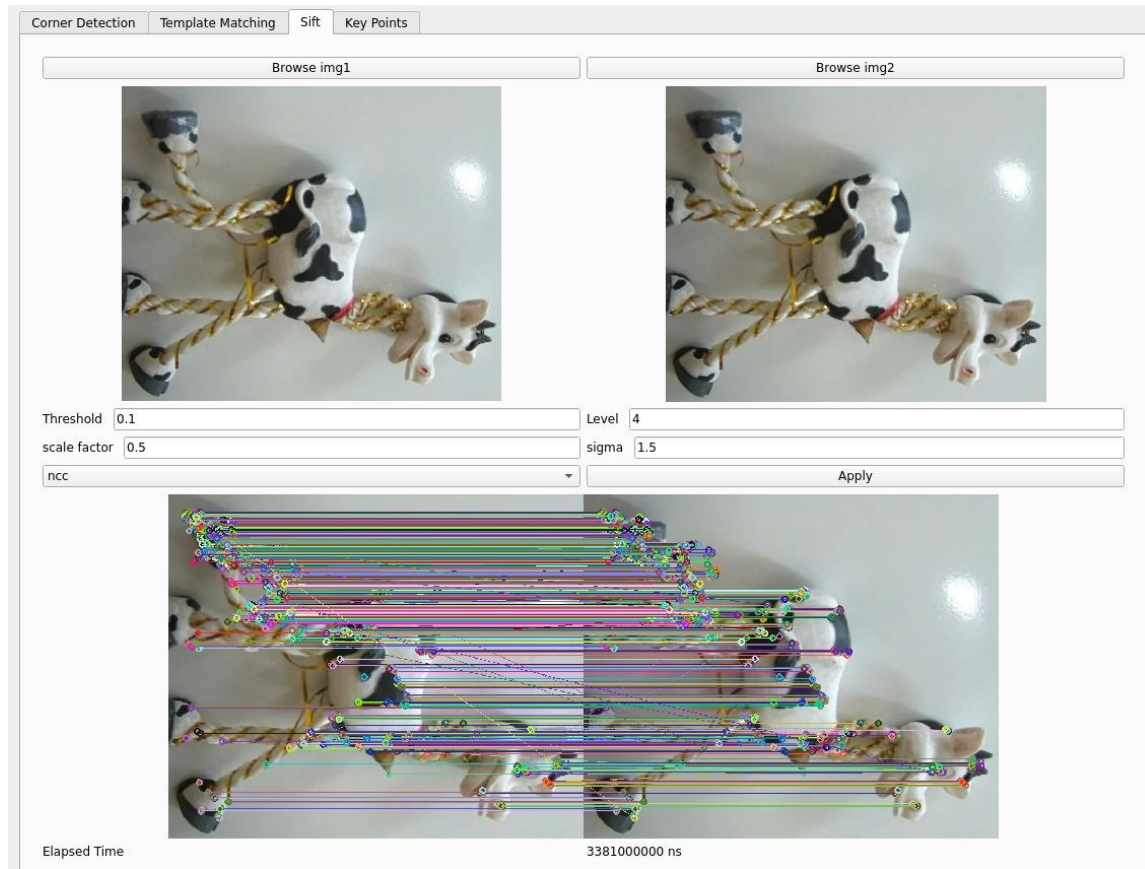


Figure 8: Image matching using SIFT(NCC).

### 3.1.3 Time:

```
Elapsed time for SIFT algorithm: 1084000000 ns.
Elapsed time for SIFT algorithm: 1084 milliseconds.
```

Figure 9: Image matching using SIFT time.

**Time:** 1084 milliseconds.

## 3.2 Template matching using similarity:

Template matching is a technique in computer vision used to locate a template image within a larger image.

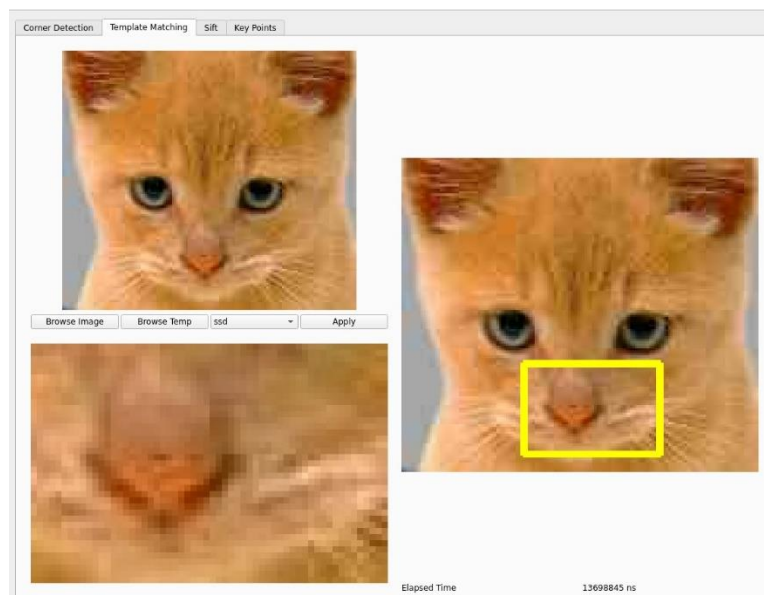
### 3.2.1 Principle:

It compares a template image with sub-regions of a larger image to find areas where the template closely matches.

### 3.2.2 Pseudocode:

1. Loop the template over the original image.
2. Calculate the similarity of the template with the part in the original image by Sum of Square Distance (SSD) or Normalized Cross Correlations (NCC).
3. Return the part of the original image with the max similarity with template.

### 3.2.3 Output image:



*Figure 10: Template matching using SSD.*



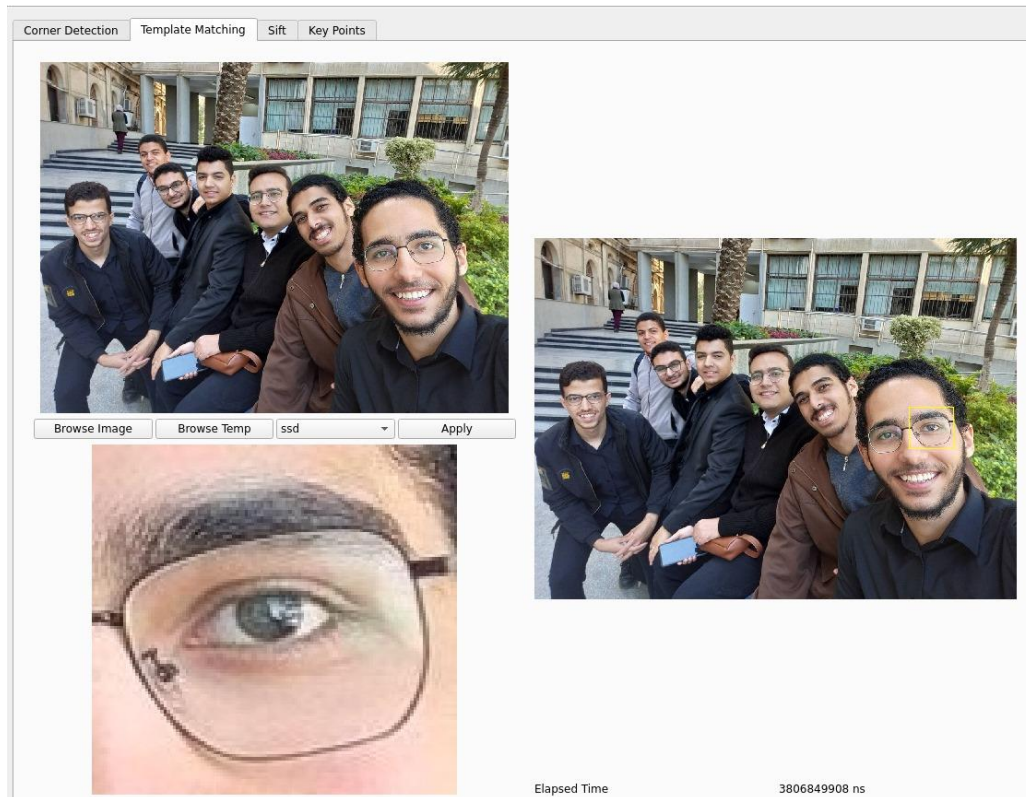


Figure 11: Template matching using SSD.

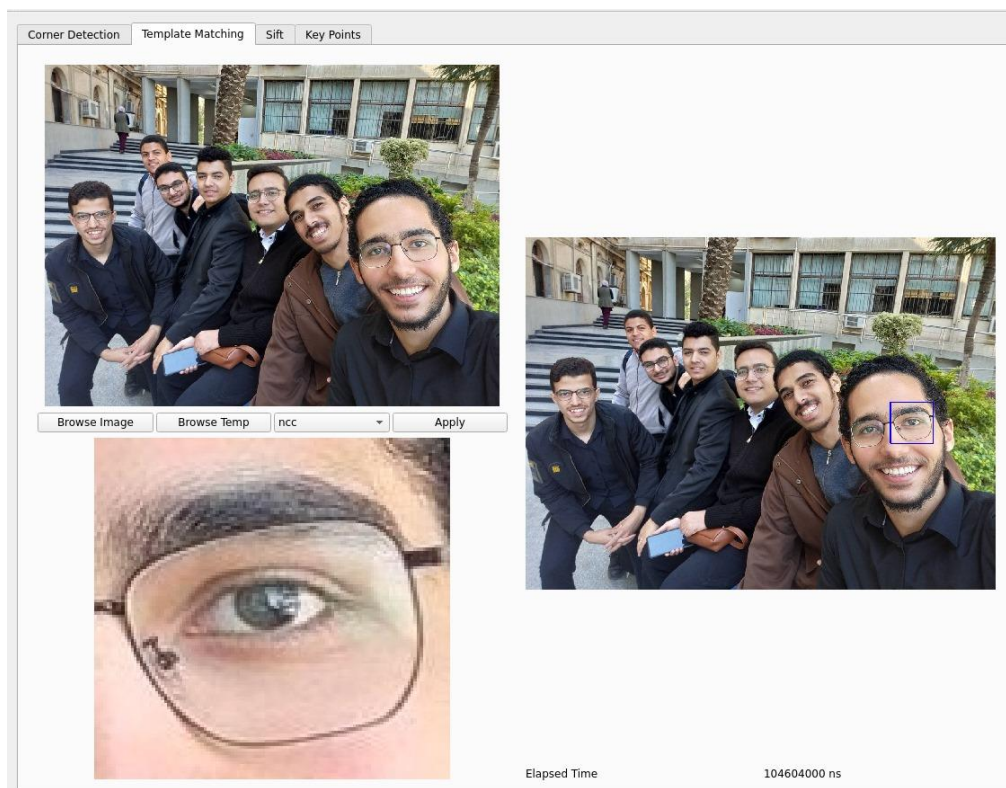
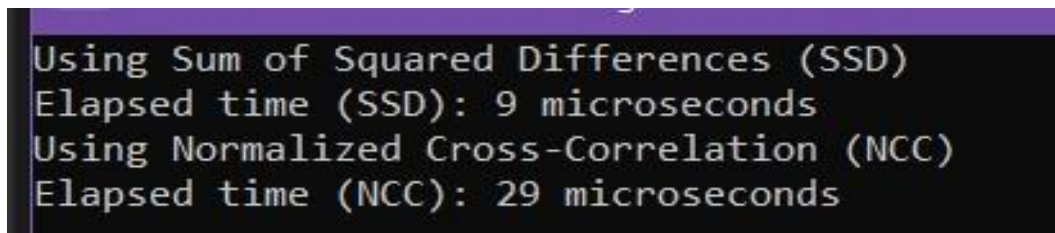


Figure 12: Template matching using NCC.

#### 3.2.4 SSD time Vs NCC time:



```
Using Sum of Squared Differences (SSD)
Elapsed time (SSD): 9 microseconds
Using Normalized Cross-Correlation (NCC)
Elapsed time (NCC): 29 microseconds
```

*Figure 13: SSD & NCC time.*

SSD time: 9 microseconds.

NCC time: 29 microseconds.