Computer Vision
(CSL7360)
Course Project Report

Object Detection

Instructor: Dr. Pratik Mazumder

Group: 47
Team members
Banoth Sri Kowshika Raj (B22CS018)
Suhani (B22CS051)
Akash Chaudhary (B22EE007)
Saurabh Chavan (B22EE061)
Unisha Tirkey (B22ES029)

Github Streamlit_Website

1. Objective

The primary objective of this project is to design and implement the Scale Invariant Feature Transform (SIFT) algorithm from the ground up for robust object detection and image matching. The project aims to achieve accurate detection of keypoints that are invariant to scale, rotation, and illumination changes. To ensure accessibility and interactivity, the system is deployed as a web application using Streamlit.

2. Methodology

2.1 Custom SIFT Pipeline

The SIFT (Scale-Invariant Feature Transform) algorithm aims to detect distinctive and repeatable keypoints from images that are invariant to scale, rotation, and illumination changes. The custom SIFT pipeline was implemented in Python, and follows these key stages:

SIFT Image Features

For any object there are many features, interesting points on the object, that can be extracted to provide a "feature" description of the object. This description can then be used when attempting to locate the object in an image containing many other objects. There are many considerations when extracting these features and how to record them. SIFT image features provide a set of features of an object that are not affected by many of the complications experienced in other methods, such as object scaling and rotation.

While allowing for an object to be recognised in a larger image SIFT image features also allow for objects in multiple images of the same location, taken from different positions within the environment, to be recognised. SIFT features are also very resilient to the effects of "noise" in the image.

The SIFT approach, for image feature generation, takes an image and transforms it into a "large collection of local feature vectors. Each of these feature vectors is invariant to any scaling, rotation or translation of the image. This approach shares many features with neuron responses in primate vision. To aid the extraction of

these features the SIFT algorithm applies a 4-stage filtering approach:

Scale-Space Extrema Detection

This stage of the filtering attempts to identify those locations and scales that are identifiable from different views of the same object. This can be efficiently achieved using a "scale space" function. Further it has been shown under reasonable assumptions it must be based on the Gaussian function. The scale space is defined by the function:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Where * is the convolution operator, $G(x, y, \sigma)$ is a variable-scale Gaussian and $I(x, y)$ is the input image.

Various techniques can then be used to detect stable keypoint locations in the scale-space. Difference of Gaussians is one such technique, locating scale-space extrema, $D(x, y, \sigma)$ by computing the difference between two images, one with scale k times the other. $D(x, y, \sigma)$ is then given by:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

To detect the local maxima and minima of $D(x, y, \sigma)$ each point is compared with its 8 neighbours at the same scale, and its 9 neighbours up and down one scale. If this value is the minimum or maximum of all these points then this point is an extrema.

Keypoint Localisation

This stage attempts to eliminate more points from the list of keypoints by finding those that have low contrast or are poorly localised on an edge. This is achieved by calculating the Laplacian , value for each keypoint found in stage 1. A method needs to be implemented for fitting a 3D quadratic function to the local sample points to determine the interpolated location of maximum. This approach uses Taylor expansion of scale space function.

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x$$

The location of extremum, is given by:

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2}\frac{\partial D}{\partial x}$$

The final offset $\hat{x}$ is added to the location of its sample point to get the interpolated estimate for the location of the extremum. The function value at the extremum, D($\hat{x}$), is useful for rejecting unstable extrema with low contrast.

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial x}\hat{x}$$

Discard keypoints with:

- Low contrast (thresholded by evaluating D(x)).
- Poor localization along edges using the **Hessian matrix**:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

and reject if:

$$\frac{(\mathrm{Tr}(H))^2}{\mathrm{Det}(H)} > \mathrm{Threshold}$$

Advantage:

- Improves the stability and accuracy of detected keypoints.
- Reduces noise and false detections, especially near edges and textured regions.

Orientation Assignment

This step aims to assign a consistent orientation to the keypoints based on local image properties. The keypoint descriptor, described below, can then be represented relative to this orientation, achieving invariance to rotation. The approach taken to find an orientation is:

- Use the keypoints scale to select the Gaussian smoothed image L, from above

- Compute gradient magnitude, *m*

$$m(x,y) = \sqrt{\big(L(x+1,y) - L(x-1,y)\big)^2 + \big(L(x,y+1) - L(x,y-1)\big)^2}$$

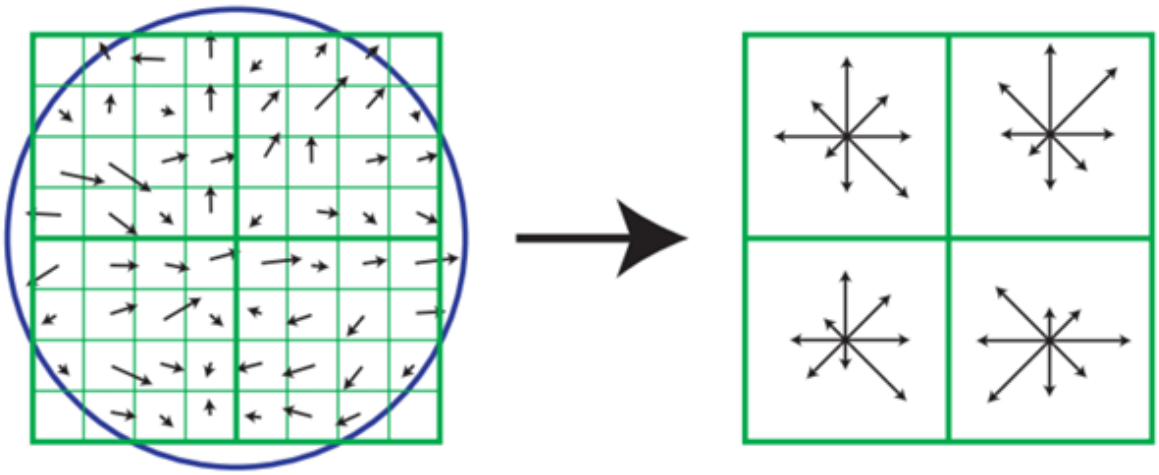
- Compute orientation, θ

$$\theta(x,y) = tan^{-1}((L(x,y+1) - L(x,y-1))/(L\big(x+1,y) - L(x-1,y)\big))$$

- Form an orientation histogram from gradient orientations of sample points
- Locate the highest peak in the histogram. Use this peak and any other local peak within 80% of the height of this peak to create a keypoint with that orientation
- Some points will be assigned multiple orientations
- Fit a parabola to the 3 histogram values closest to each peak to interpolate the peaks position

Keypoint Descriptor

The descriptor for the local image region is highly distinctive yet is as invariant as possible to remaining variations, such as change in illumination or 3D viewpoint



A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window. Then, these samples are put together to make orientation histograms that show the contents

over 4x4 subregions. The length of each arrow shows the sum of the gradient magnitudes near that direction in the region.

- Extract a 16×16 region around the keypoint.
- Divide into 16 blocks (4×4 regions).
- For each block, compute an 8-bin orientation histogram based on gradient magnitude and direction.
- Concatenate all 16 histograms into a 128-dimensional descriptor vector (16 × 8 = 128).
- Normalize the descriptor vector to unit length to reduce the influence of illumination.
- Optionally, threshold large values and renormalize to improve robustness to affine illumination changes.

Advantage:

- Highly distinctive – suitable for matching across different images.
- Invariant to rotation, scale, illumination, and minor affine transformations.
- Compact (128 floats per keypoint) and efficient to compare.

<u>Keypoint Matching</u>

It Identify **correspondences between images** using the generated descriptors.

Working:

- For each keypoint descriptor in image A, find the nearest neighbour in image B using **Euclidean distance**:

$$d(p,q) = \sqrt{\sum_{i=1}^{128}(p_i - q_i)^2}$$

- Use **Lowe's Ratio Test** to eliminate ambiguous matches:

$$\frac{d_1}{d_2} < 0.75$$

where d1 and d2 are distances to the closest and second-closest matches respectively.

- Matches passing the test are likely to be correct and reliable.

Advantage:

- Robust matching using distance ratio minimizes false positives.
- Can be used for image stitching, object recognition, tracking, etc.

2.2 OpenCV SIFT Integration

To benchmark performance and correctness, OpenCV's native SIFT implementation was used. Comparative analysis included:

Number of Detected Keypoints:

- Checks whether the custom method captures a similar number of meaningful features as OpenCV.
- Too few → loss of information; too many → noisy/irrelevant keypoints.

Matching Accuracy:

- Evaluate match correctness by visual inspection or geometric constraints (e.g., RANSAC).
- Custom implementation's descriptors are expected to yield comparable matches to OpenCV's.

Execution Time:

- Important for real-time systems or embedded applications.
- Custom implementation helps understand internal mechanics despite being slower.

2.3 Deployment

A user-friendly web interface was developed using Streamlit. Users can upload two images and visualize keypoints and mappings.

3. Results

- The implementation accurately detects keypoints and matches them across different images.

- A total of **39 valid matches** were obtained using the custom pipeline.
- Execution time for the custom method is higher but demonstrates all SIFT principles effectively.
- The Streamlit deployment provides an accessible way to test and demonstrate the functionality interactively.

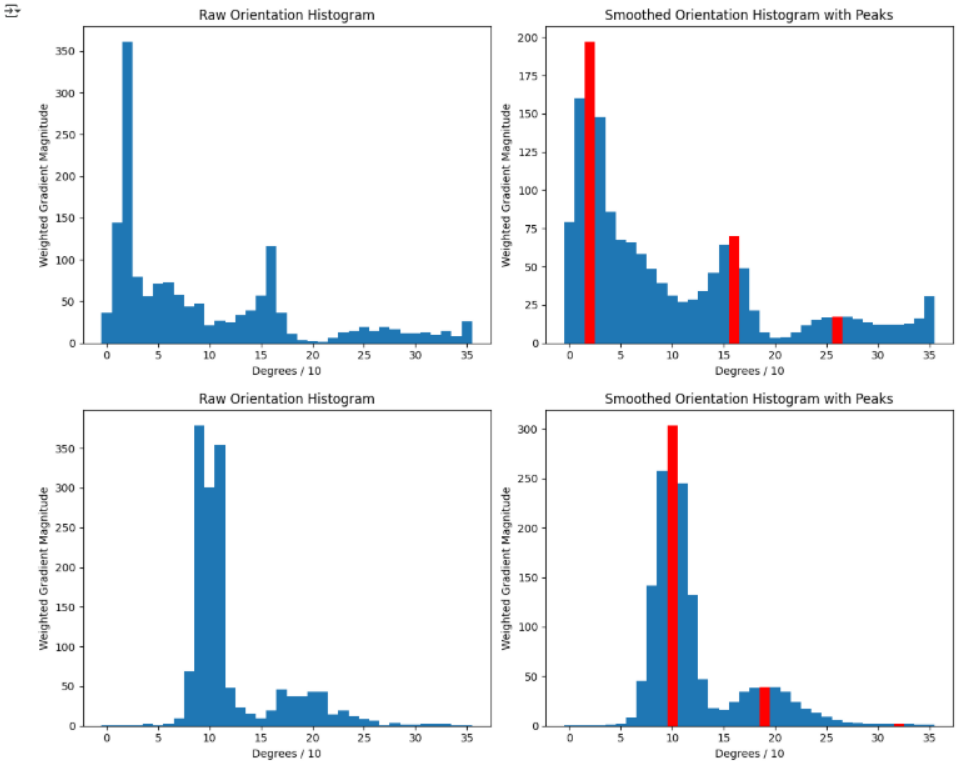

```
img1 = cv2.imread('parleg.jpg', 0)     # queryImage
```

Original image

Gaussian filter on original image

After implementing the SIFT algorithm on above preprocessed images, we can identify the number of key points in both the images as follows:

```
Number of keypoints in image 1 with SIFT from scratch :  535
Number of keypoints in image 2 with SIFT from scratch :  658
```

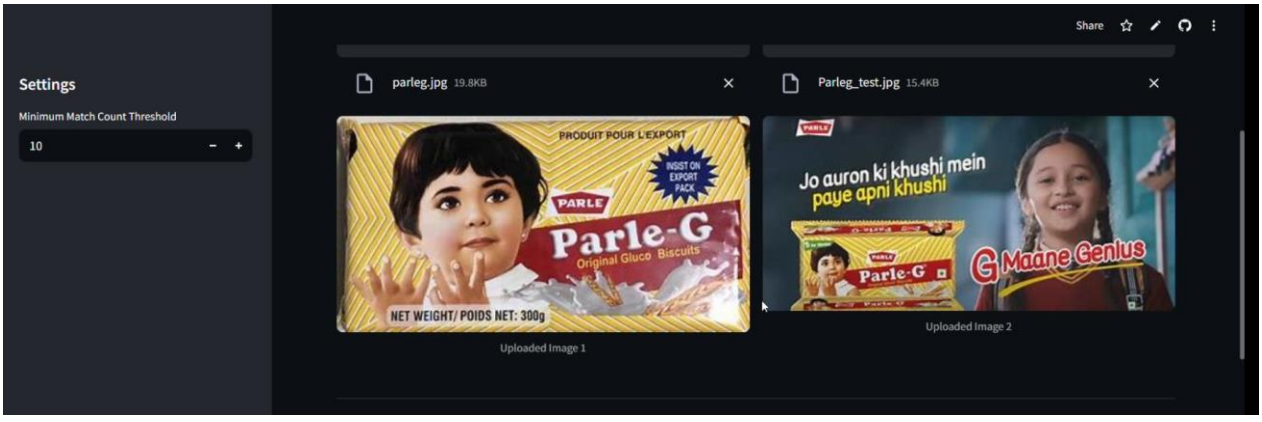


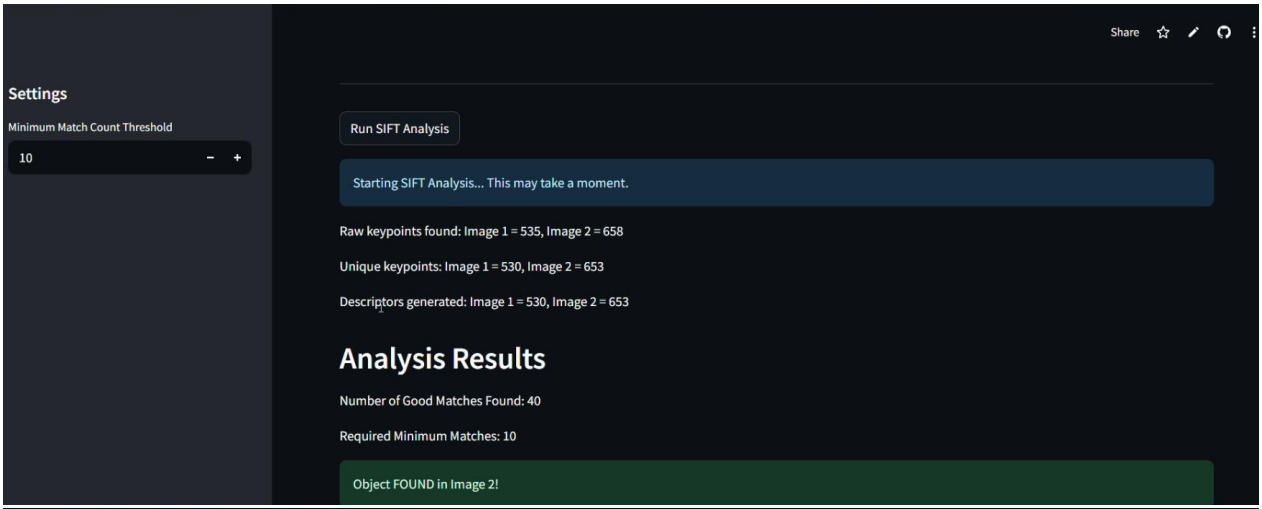

```
MIN_MATCH_COUNT = 10

template_matching(img1, img2, img1_color, img2_color, keypoints1, keypoints2, descriptors1, descriptors2, MIN_MATCH_COUNT = 10, object_found = False)
```

```
Number of Matches with SIFT from scratch :  39
```

Results from Streamlit deployment:





Visual Results Include:

- Gaussian and DoG pyramid visualizations
- Keypoints identified on individual images
- Keypoint matches between image pairs
- Orientation histograms and descriptor vectors

4.Individual Contributions

- **Banoth Sri Kowshika Raj (B22CS018):**
  Designed and implemented the Gaussian and DoG filtering stages. Developed the image pyramid and contributed to keypoint detection logic.

- **Suhani (B22CS051):**
  Led the deployment of the Streamlit-based web application. Structured the report, created demonstration scripts. Assisted with application testing and documentation.

- **Akash Chaudhary (B22EE007):**
  Contributed to the implementation of the template matching algorithm.

- **Saurabh Chavan (B22EE061):**
  Keypoint detection, orientation histogram. Streamlit deployment, demonstration video.

- **Unisha Tirkey (B22ES029):**
  Worked on keypoint descriptor construction and orientation assignment. , and prepared the final presentation. Assisted in testing the web application.

References:

1. lowe-ijcv04.pdf http://www.cs.berkeley.edu/%7Emalik/cs294/lowe-ijcv04.pdf
2. The University of Edinburgh : SIFT Image Features
3. TU Delft Repository: Object detection using SIFT. https://repository.tudelft.nl/record/uuid%3Ad43d8df5-6cc5-429f-9438-cd0d3c5b480f
4. Michieli, U. (n.d.). Object Recognition with SIFT, SURF and LBP. https://umbertomichieli.github.io/download/course_projects/Computer_Vision.pdf

5. Chalmers University of Technology: SIFT for Traffic Signs. https://webfiles.portal.chalmers.se/s2/undergraduate/SSY095/PDF/6.1.1SIFTForTrafficSigns.pdf