Faculty of Engineering
Cairo University

Cairo University

Computer Vision

Task-3

Team 19

Name	Section	B.N
Mohamed Alaa Ali	2	19
Mohamed Ibrahim Ismail	2	9
Mohamed El-sayed Ali	2	10
Mohamed El-sayed Eid	2	11

Table of Contents

Introduction:

In the realm of image processing and computer vision, the extraction and comparison of distinctive features from visual data are essential for various applications. This study focuses on analyzing a set of grayscale and color images, aiming to extract unique features and match them for further analysis. Tasks include feature extraction using the Harris operator and SIFT, as well as feature matching using SSD and normalized cross-correlation. By recording computation times, we aim to evaluate the efficiency and efficacy of these methodologies in handling diverse image datasets.

Tasks To Implemented:

1. Harris operator feature extraction
2. SIFT feature descriptor generation
3. Feature matching using SSD and normalized cross-correlation
4. Computation time tracking for matching

1. Harris operator and lambda-minus

1.1 Harris operator

The Harris corner detector algorithm is a widely used method in computer vision for detecting corners or significant points within an image. It operates by analyzing variations in intensity in small neighborhoods of the image and identifies regions where these variations are large in multiple directions, indicating the presence of corners or edges. This algorithm is particularly valuable in tasks such as feature extraction, image registration, and object recognition.

- Step 1: Convert image to grayscale
  gray Image = ConvertToGrayscale(image)

- Step 2: Apply Gaussian blur for smoothing
  smoothed Image = Gaussian Blur (gray Image)

- Step 3: Compute gradients using Sobel operator
  $Sobel_X$, $Sobel_Y$ = Compute Gradients (smoothed Image)

- Step 4: Compute products of derivatives
  I^2_x, I^2_y, I_{xy} = ComputeProductsOfDerivatives ($Sobel_X$, $Sobel_Y$)

- Step 5: Apply Gaussian smoothing to the products
  $sumI^2_x$, $sumI^2_y$, $sumI_{xy}$ = Apply Smoothing (I^2_x, I^2_y, I_{xy})

- Step 6: Compute Harris response
  Harris Response = Compute Harris Response ($sumI^2_x$, $sumI^2_y$, $sumI_{xy}$, k)

- Step 7: Thresholding and get corners
  corners = Threshold and Extract Corners (Harris Response, threshold)

- Step 8: Draw corners on the original image
  result Image = Draw Corners (image, corners)

return **result Image**

1.2 λ-minus

Lambda-minus (λ^-) refers to a variant of the Harris corner detection algorithm that adjusts the sensitivity of corner detection. It involves tweaking the response function by subtracting a fraction of the trace of the auto-correlation matrix. This adjustment helps in enhancing corner detection in cases where the image has low contrast or when the corners

are located at edges. By fine-tuning the response function, lambda-minus can improve the robustness and accuracy of corner detection in various scenarios, making it a valuable enhancement to the traditional Harris corner detection algorithm.

1. **Preprocessing:**
   - If the image is in color, convert it to grayscale.
   - Apply Gaussian blur to the grayscale image to reduce noise and smoothen edges.

2. **Compute Gradients:**
   - Use the Sobel operator to compute gradients along the X and Y directions.

3. **Compute Lambda-Minus:**
   - For each pixel location in the gradient images:
     - Compute the structure tensor H using the gradients.
     - Find the eigenvalues of H and determine the minimum eigenvalue
     - Create a matrix storing the minimum eigenvalue (λ^-) at each pixel location.

4. **Calculate Threshold:**
   - Determine a threshold value based on a specified quantile of the absolute values of the minimum eigenvalue matrix.

5. **Detect Corners:**
   - Identify corners as pixel locations where the absolute value of the minimum eigenvalue exceeds the threshold.

6. **Draw Corners:**
   - Draw circles at the identified corner locations on the original image.

7. **Return Result:**
   - Return the image with drawn corners.

1.3 Computation time

1.3.1 Harris corner detection:

- M represents the height (number of rows) of the image.
- N represents the width (number of columns) of the image.
- k represents the size of the Gaussian kernel

- For the Harris corner detector algorithm:
  1. Gaussian Blur:
     - Total multiplications: $O(M * N * k^2)$
     - Total additions: $O(M * N * (k^2 - 1))$
  2. Compute Harris Response:
     - Total multiplications: $O(M * N)$
     - Total additions: $O(M * N)$

1.3.2 Lambda-minus:

- Total multiplications: $O(M * N * a) + O(M * N * c)$
- Total additions: $O(M * N * b) + O(M * N * d)$

1.4 Results of Harris

2. Scale Invariance Feature Transform

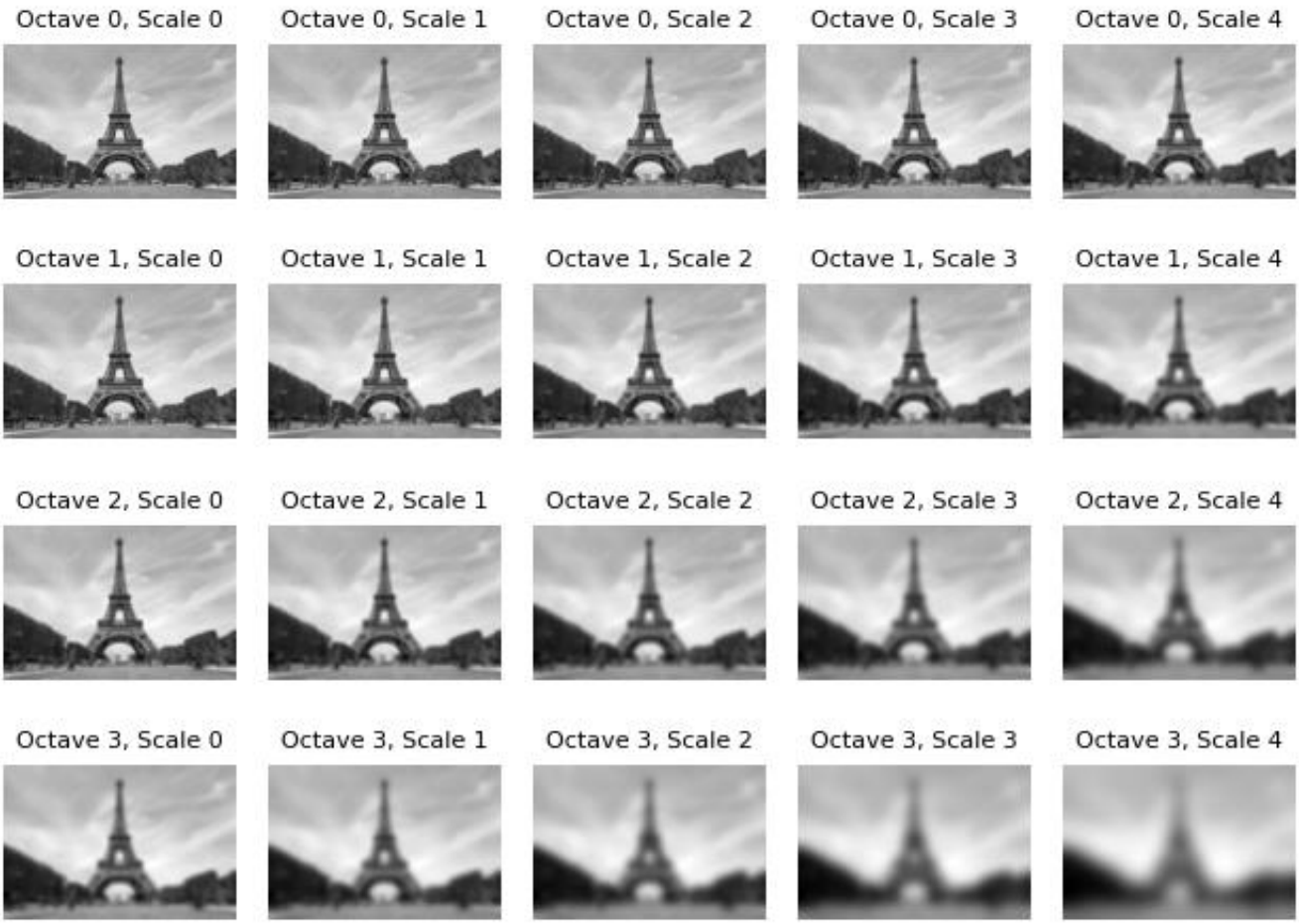
SIFT (Scale-Invariant Feature Transform) is a robust and widely used algorithm for detecting and describing distinctive local features in images. It operates by identifying key points in an image that are invariant to scale, rotation, and illumination changes. These key points are then described using histograms of local image gradients, creating feature vectors that are highly distinctive and invariant to various transformations.

SIFT consists of several key steps:

2.1 Scale-space Extrema Detection: The algorithm identifies potential key points by searching for local extrema in scale-space, using difference-of-Gaussian (DoG) images to detect stable key points across different scales.

2.1.1 Constructing a scale space

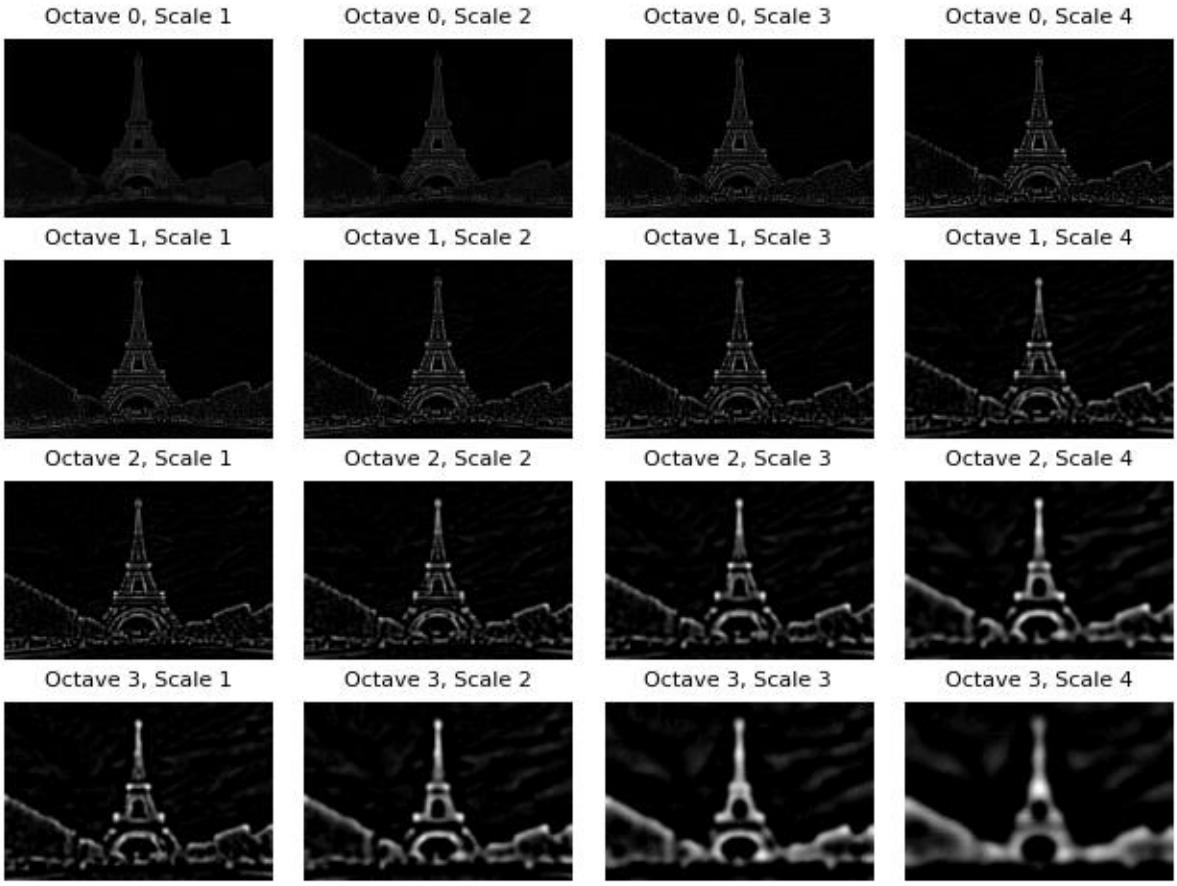
To construct a scale space, we begin with the original image and progressively apply Gaussian blur to generate a series of increasingly blurred versions. This process allows us to selectively emphasize certain objects while suppressing the influence of background elements in the scene.



2.1.2 Difference of Gaussians (DoG)

Utilizing the scale space obtained from the previous step:

- We compute the difference between two adjacent scales.
- These Difference of Gaussian (DoG) images serve as effective tools for identifying significant key points within the image.

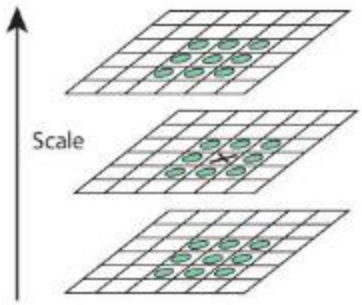


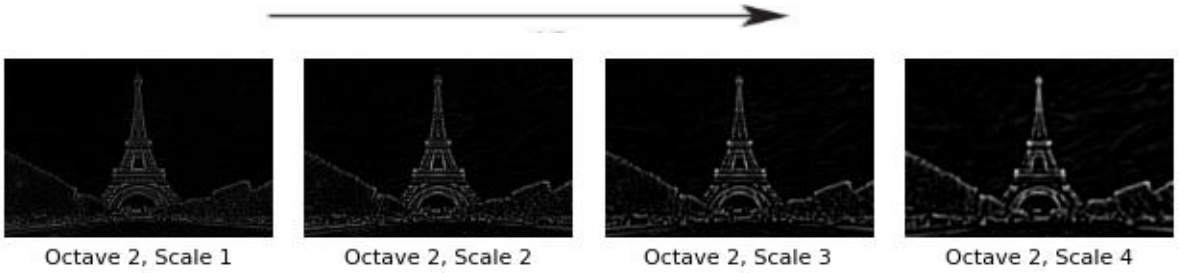
2.2 Key point Localization: After identifying potential key points, SIFT refines their locations to subpixel accuracy by fitting a quadratic function to the nearby pixel intensities.

2.2.1 Finding Local Extrema

- Each pixel is compared with 26 neighbors.
- 9 from upper scale +9 from lower scale +8 from current scale
- Take if maximum or minimum.

- For same image, it is not necessary for its corners to be localized at same scale.
- Ignore first and last DoG (no enough neighbors)

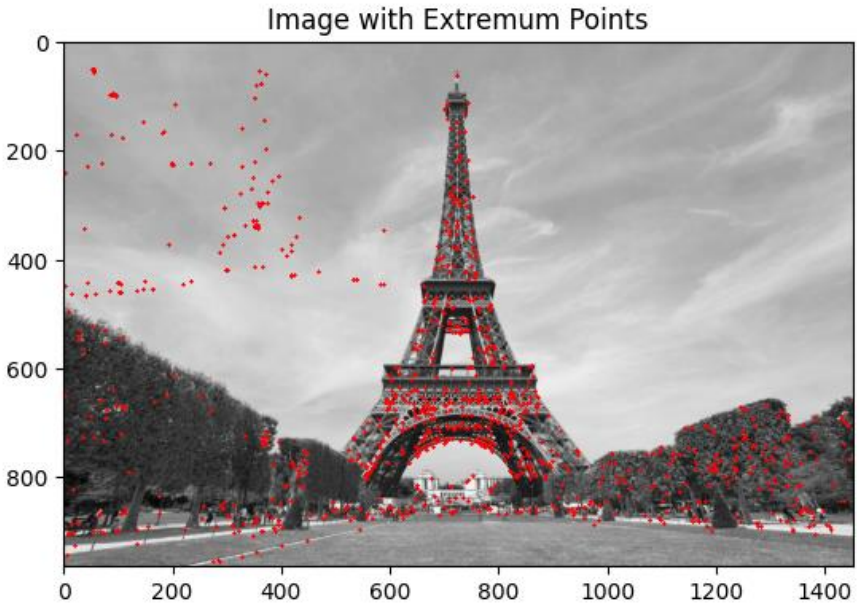




2.2.2 Eliminate Low contrast and Edges

Reject points with bad contrast:

- DoG smaller than 0.03
- Reject edges



2.3 Orientation Assignment: Each key point is assigned a dominant orientation based on local image gradients, ensuring invariance to image rotation.

2.4 Descriptor Generation: SIFT constructs a local feature descriptor for each key point by capturing the distribution of gradient orientations in its neighborhood. These descriptors are highly distinctive and robust to various image transformations.

3. Feature Matching

Feature matching is used in identification of corresponding points between images. Here will show the implementation of feature matching techniques, focusing on the Sum of Squared Differences (SSD) and Normalized Cross-Correlation (NCC) methods.

3.1 Algorithm:

- **Input:**
  - image1: First input image.
  - image2: Second input image.
  - feature matching method: either SSD or NCC

- **Detect Key points and Descriptors Method:**
  Utilize the SIFT Detector class to compute key points and descriptors for both input images.

- **SSD Method:**
  Compute the Sum of Squared Differences (SSD) between descriptors to measure feature similarity.

$$SSD(I_1, I_2) = \sum_{x,y} (I_1(x,y) - I_2(x,y))^2$$

- **NCC Method:**
  Compute the Normalized Cross-Correlation (NCC) between descriptors to measure feature similarity.

$$NCC(I_1, I_2) = \frac{\Sigma_{xy} (I_1(x,y) * I_2(x,y))}{\sqrt{(\Sigma_{xy} I_1(x,y)^2)} * \sqrt{(\Sigma_{xy} I_2(x,y)^2)}}$$

- **Match Key points:**
  Match key points between the two images based on the chosen method (SSD or NCC) and a specified threshold.

- **Get Key point Coordinates:**
  Extract the coordinates of matched key points from the key points list.

- **Draw Matches:**
  Draw lines between matched key points on the input images to visualize the matches.

- **Return Output:**
  Return an image containing both input images with matching lines indicating corresponding key points.

3.2 Computation time

- N_1: Number of key points detected in image 1.
- N_2: Number of key points detected in image 2.
- k: Size of the descriptor vectors.

3.2.1 SSD:

- Total count of multiplications: $O(\min(N_1, N_2) \times \min(N_1, N_2) \times k)$.
- Total count of additions: $O(\min(N_1, N_2) \times \min(N_1, N_2) \times (k - 1))$.

3.2.2 NCC:

- Total count of multiplications: $O(\min(N_1, N_2) \times \min(N_1, N_2) \times k)$.
- Total count of additions: $O(\min(N_1, N_2) \times \min(N_1, N_2) \times k)$.