- Lab 06: Feature Detection and Description (Detectors & Descriptors)
- Objective:
 - Understand and implement feature detection and description.
 - Use Harris, SIFT, and ORB to detect keypoints and extract descriptors.
 - Visualize and compare different feature detection results.

★ Tools & Libraries:

- Python 3.x
- OpenCV
- NumPy
- Matplotlib (optional)
- **Dataset:**

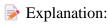
Use standard grayscale or color images like lena.jpg, building.jpg, box.png, or box_in_scene.png.

- **E** Complete Lab Code with Explanations:
 - 1. Import Required Libraries

import cv2

import numpy as np

import matplotlib.pyplot as plt



We import OpenCV for computer vision tasks, NumPy for array manipulation, and Matplotlib to optionally display images using plots.

2. Load and Convert Image to Grayscale

```
img = cv2.imread('box.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Explanation:

We load the input image in color, and then convert it to grayscale since most feature detection algorithms work on single-channel intensity images.

3A. Harris Corner Detection

```
harris = cv2.cornerHarris(np.float32(gray), blockSize=2, ksize=3, k=0.04)
harris = cv2.dilate(harris, None) # to mark the corners more clearly
img_harris = img.copy()
```

 $img_harris[harris > 0.01 * harris.max()] = [0, 0, 255]$

Explanation:

- cornerHarris detects corners based on gradient change in intensity.
- blockSize is the neighborhood size considered for corner detection.
- ksize is the aperture parameter for the Sobel operator.
- k is the Harris detector free parameter (usually between 0.04–0.06).
- We dilate the response image for better visualization.
- Detected corners are marked in red on the copy of the original image.

3B. SIFT – Scale-Invariant Feature Transform

```
sift = cv2.SIFT_create()
kp_sift, des_sift = sift.detectAndCompute(gray, None)
img_sift = cv2.drawKeypoints(img, kp_sift, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
```

Explanation:

- SIFT is a scale- and rotation-invariant detector and descriptor.
- detectAndCompute returns keypoints and descriptors.
- drawKeypoints visualizes keypoints;
 DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS draws orientation and scale.

3C. ORB - Oriented FAST and Rotated BRIEF

```
orb = cv2.ORB_create()
kp_orb, des_orb = orb.detectAndCompute(gray, None)
img_orb = cv2.drawKeypoints(img, kp_orb, None, color=(0, 255, 0), flags=0)
```

Explanation:

- ORB is a fast, efficient alternative to SIFT/SURF (and free of patent restrictions).
- It combines the FAST keypoint detector and BRIEF descriptor with orientation and scale handling.
- We draw keypoints in green.
- 4. Display Results

```
cv2.imshow('Harris Corners', img_harris)
cv2.imshow('SIFT Keypoints', img_sift)
```

```
cv2.imshow('ORB Keypoints', img_orb)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Explanation:

- Shows the output images with visualized keypoints.
- waitKey(0) waits for a key press to close windows.
- 5. (Optional) Feature Matching using ORB

```
img1 = cv2.imread('box.png', 0)
img2 = cv2.imread('box_in_scene.png', 0)
orb = cv2.ORB_create()
kp1, des1 = orb.detectAndCompute(img1, None)
kp2, des2 = orb.detectAndCompute(img2, None)

# Brute-force matcher with Hamming distance
bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
matches = bf.match(des1, des2)
matches = sorted(matches, key=lambda x: x.distance)

# Draw top 20 matches
matched_img = cv2.drawMatches(img1, kp1, img2, kp2, matches[:20], None, flags=2)
cv2.imshow("ORB Feature Matching", matched_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Explanation:

- Uses ORB to detect and describe features in two images.
- Brute-Force matcher compares descriptors using Hamming distance.
- The best 20 matches are drawn on a combined image for visual comparison.

Output:

- Three windows displaying keypoints detected by Harris, SIFT, and ORB.
- (Optional) Matching visualization between two images.

B Lab Report Checklist:

- Objective and tools used.
- Code with explanations.
- Screenshots of results for each method.