جامعة الأمير مقرن بن عبد العزيز
Prince Mugrin University

Lab 6

SIFT Detector and Features Matching with OpenCV

Learning Objectives:

- Understand how SIFT works and why it is useful.
- Detect keypoints and compute descriptors using SIFT.
- Perform feature matching between two images..
- Use SIFT for applications like object recognition and image stitching

Introduction

Feature detection and matching are fundamental techniques in computer vision used to identify corresponding points in different images. These techniques enable tasks such as **object recognition, image stitching, 3D reconstruction, augmented reality, and autonomous navigation**.
Among the most powerful feature detection methods is the **Scale-Invariant Feature Transform (SIFT)** algorithm, which extracts unique keypoints and descriptors that remain consistent across different scales, rotations, and lighting conditions.

1️⃣ Understanding SIFT (Scale-Invariant Feature Transform)

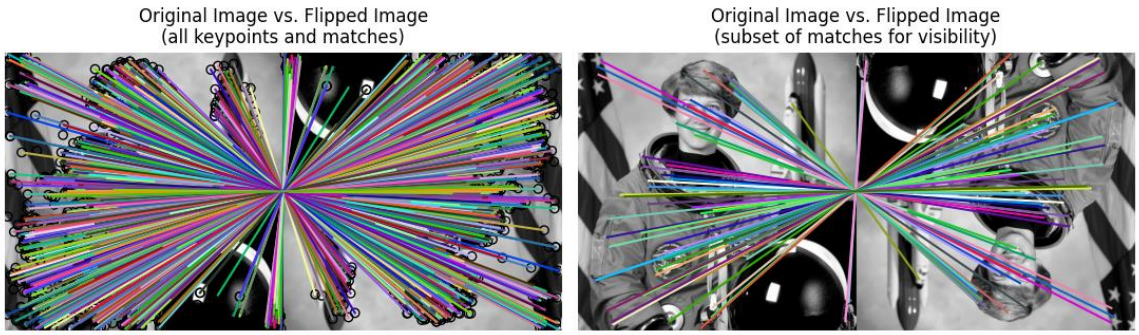
📌 Why Is SIFT Important?

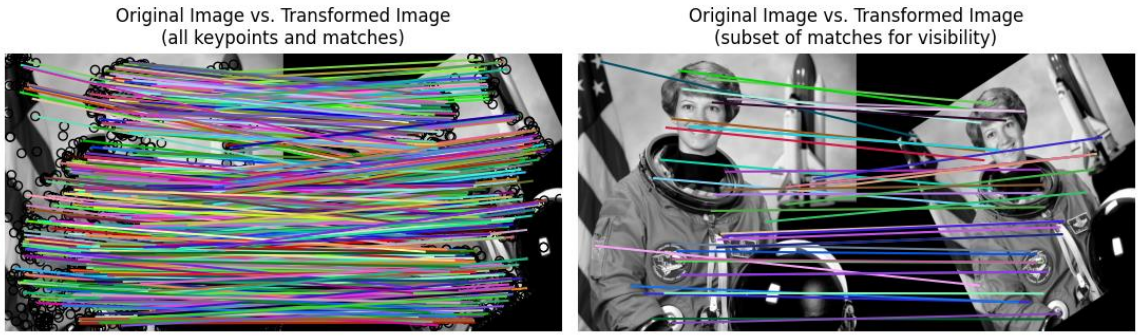
SIFT is a widely used method for detecting and describing local features in images. Unlike traditional edge detection methods, SIFT keypoints are:

- **Scale-invariant:** They remain consistent regardless of the image size.
- **Rotation-invariant:** They can be detected in different orientations.
- **Illumination-robust:** They work well under varying lighting conditions.

📌 How Does SIFT Work?

- **Scale-space extrema detection:** Identifies keypoints at different scales.
- **Keypoint localization:** Filters out low-contrast keypoints.
- **Orientation assignment:** Ensures keypoints are rotation-invariant.
- **Descriptor computation:** Creates a feature vector for each keypoint.





2️⃣ Detecting SIFT Features

🛠️ Task: Extract SIFT Keypoints from an Image

```
import cv2
import numpy as np

# Load an image
gray = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Initialize SIFT detector
sift = cv2.SIFT_create()

# Detect keypoints and compute descriptors
keypoints, descriptors = sift.detectAndCompute(gray, None)

# Draw keypoints on the image
       img_keypoints = cv2.drawKeypoints(gray, keypoints, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

cv2.imshow('SIFT Keypoints', img_keypoints)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

3️⃣ Understanding Feature Matching

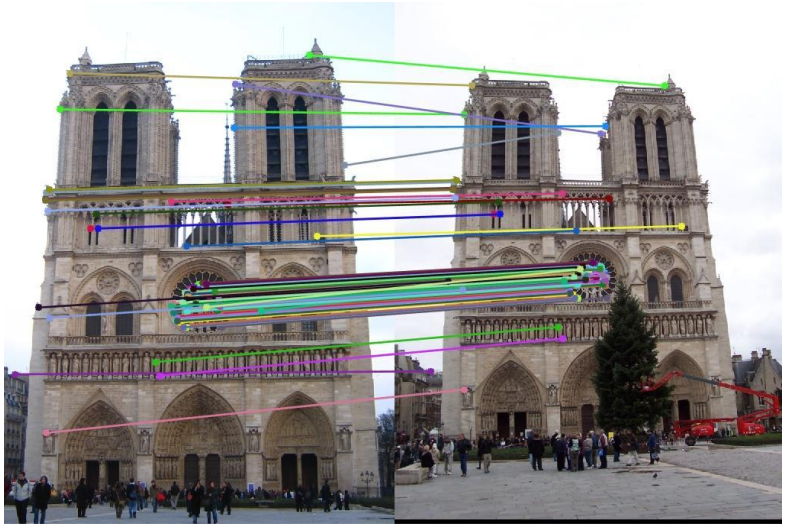
📌 Why Is Feature Matching Important?

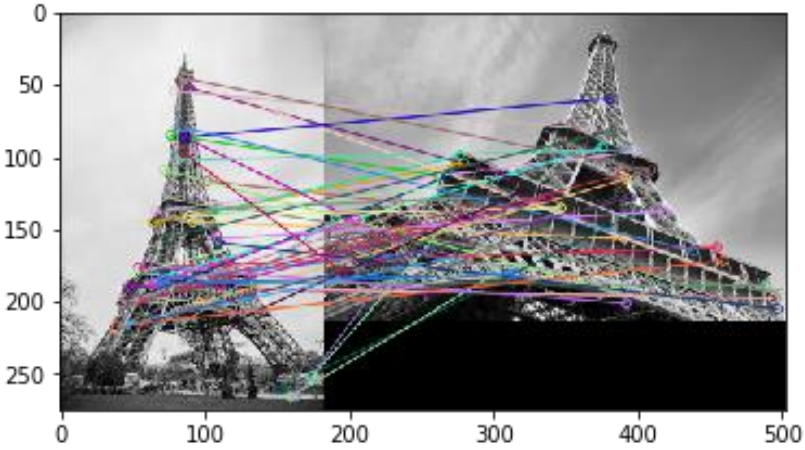
Feature matching allows us to compare images and recognize patterns despite changes in **scale, rotation, illumination, and perspective**. It enables:

- **Object recognition:** Identifying objects in different images.
- **Image stitching:** Combining multiple images to create panoramas.
- **3D reconstruction:** Estimating depth and structure from multiple images.
- **Augmented reality (AR):** Detecting real-world surfaces to overlay virtual content.
- **Autonomous navigation:** Enabling robots and self-driving cars to recognize landmarks and objects.

📌 How Does Feature Matching Work?

1. **Feature Detection:** Identify keypoints in an image (e.g., edges, corners, textures).
2. **Descriptor Computation:** Generate unique numerical representations of each keypoint.
3. **Feature Matching:** Compare descriptors between two images to find corresponding features.
4. **Filtering Matches:** Apply techniques like Lowe's ratio test to remove weak matches.





4️⃣ Feature Matching with SIFT

🛠️ Task: Match Features Between Two Images

```
# Load two images
gray1 = cv2.imread('image1.jpg', cv2.IMREAD_GRAYSCALE)
gray2 = cv2.imread('image2.jpg', cv2.IMREAD_GRAYSCALE)

# Detect and compute features
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(gray1, None)
kp2, des2 = sift.detectAndCompute(gray2, None)

# Use BFMatcher for feature matching
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1, des2, k=2)

# Apply ratio test (Lowe's ratio test)
good_matches = []
for m, n in matches:
    if m.distance < 0.75 * n.distance:
        good_matches.append(m)

# Draw matches
img_matches = cv2.drawMatches(gray1, kp1, gray2, kp2, good_matches, None,
flags=cv2.DrawMatchesFlags_NOT_DRAW_SINGLE_POINTS)

cv2.imshow('SIFT Feature Matching', img_matches)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

🛠️ Exercise 1: Extract SIFT Keypoints from an Image

Objective:

Extract and visualize SIFT keypoints from an image to understand how local features are detected.

Instructions:

1. *Load an image and convert it to grayscale.*
2. *Use SIFT to detect keypoints and compute descriptors.*
3. *Draw the detected keypoints on the image and display it.*

🛠️ Exercise 2: Match Features Between Two Images

Objective:

Match features between two images using SIFT and Brute Force Matcher to find correspondences.

Instructions:

1. *Load two images and convert them to grayscale.*
2. *Detect and compute SIFT features in both images.*
3. *Use Brute Force Matcher (BFMatcher) to find feature correspondences.*
4. *Filter matches using Lowe's ratio test.*
5. *Draw the matching keypoints between the two images and display them.*

References:

- **OpenCV**
- **Good Article**

- **Youtube**