Name: Prajwal Ravindra Awate

Batch: A_48 **CV (TA – 2) Mini Project**

Image Processing Algorithms Implementation

Title: Implementation of Key Feature Detection and Matching Algorithms using OpenCV

Objective:

The objective of this project is to implement and visualize feature detection and matching algorithms on real-world images using Python and OpenCV. The focus is on:

- Detecting keypoints and descriptors using **SIFT**.
- Eliminating outlier matches using **RANSAC**.
- Detecting corners in grayscale images using the **Harris Corner Detector**.

---

1. SIFT (Scale-Invariant Feature Transform)

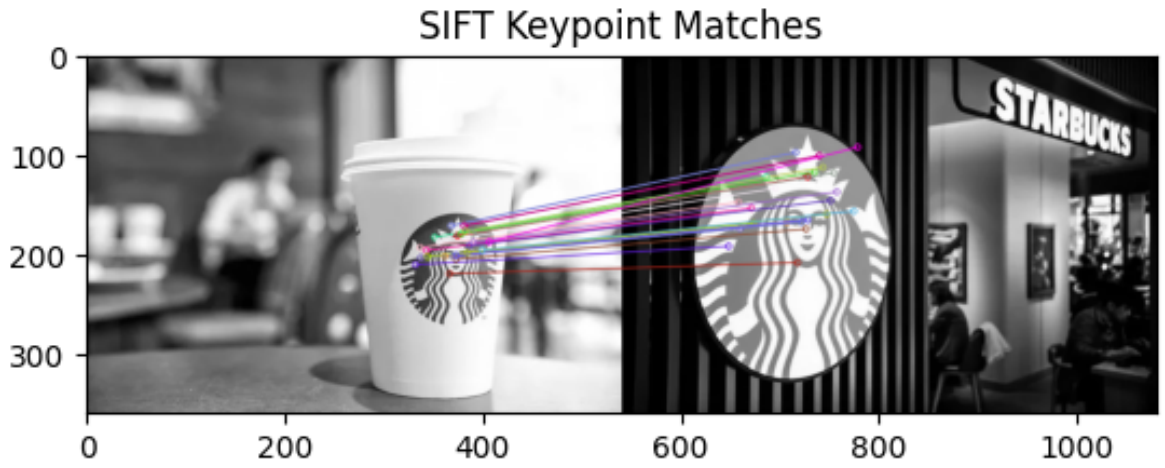
Purpose:
To detect and describe distinctive keypoints in images that are invariant to scale, rotation, and illumination.

How it works:

- SIFT detects keypoints by identifying local extrema in scale space.
- Each keypoint has a descriptor—a vector describing its appearance.
- Matching is done using descriptor similarity (e.g., via Brute-Force Matcher).

Use Case:
Recognizing the same object in two different photos.

Output Image:



Observation: You can see green lines connecting matching features (edges, corners, textures) in both images. It shows how SIFT can find corresponding points even if the images differ in angle or scale.

---

2. RANSAC (Random Sample Consensus)

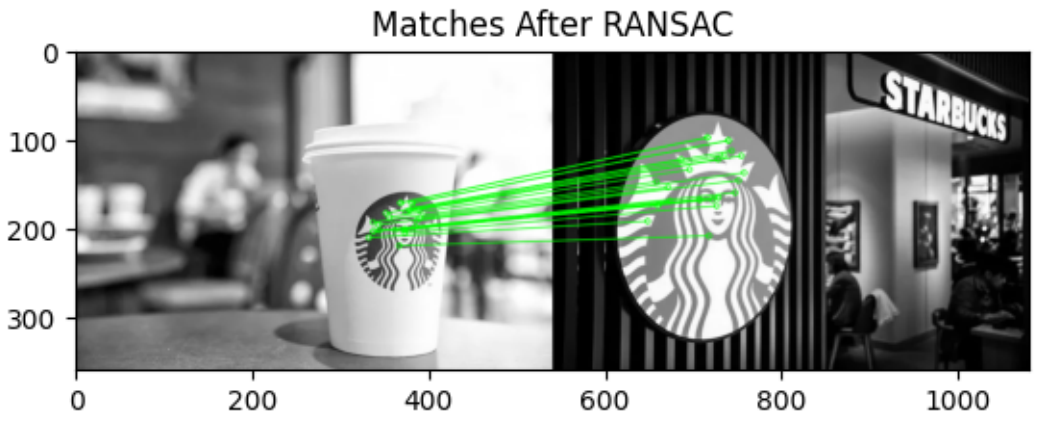
Purpose:
To estimate a transformation model (e.g., homography) between two sets of points, while filtering out outliers.

How it works:

- RANSAC repeatedly selects random subsets of matches.
- It computes a model and checks how many points fit well (inliers).
- The best-fitting model with the most inliers is selected.

Use Case:
Filtering poor keypoint matches and aligning two overlapping images for panorama stitching

Output:



Observation:

Compared to the SIFT output, many bad matches are removed. Only accurate feature correspondences remain — green lines now connect well-aligned parts (RANSAC removes noise/outliers).

---

3. Harris Corner Detector

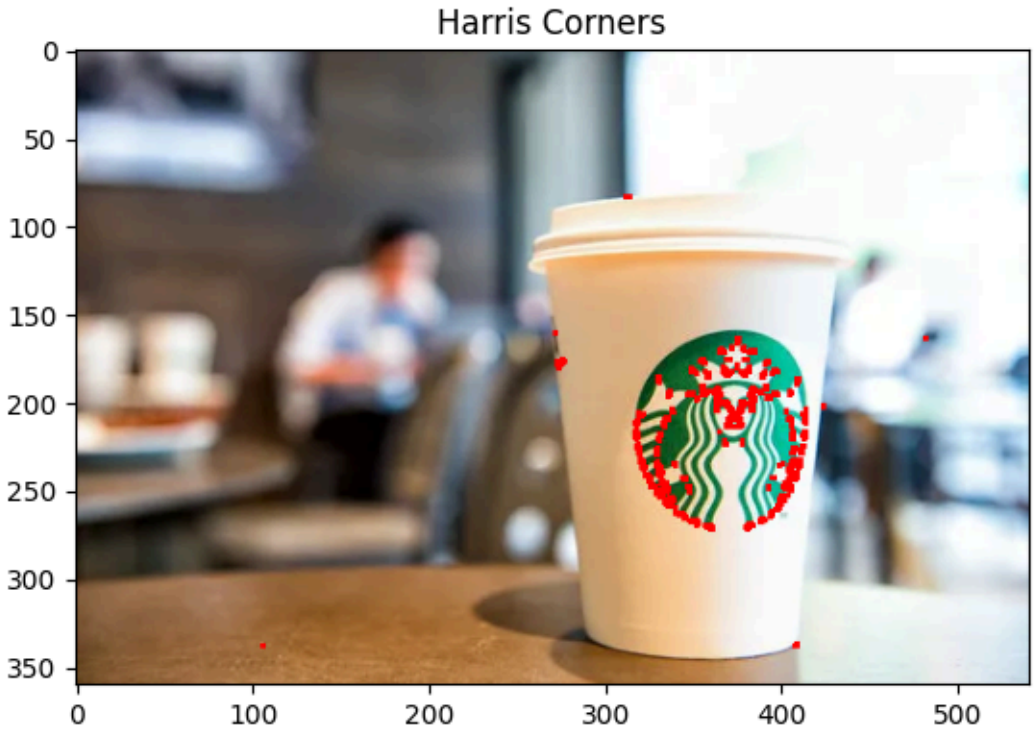
Purpose:
To detect corners in an image, which are useful for tracking and mapping features.

How it works:

- Calculates image gradients and analyzes local changes in intensity.
- A response function detects regions with high variations in all directions—typically corners.

Use Case:
Detecting features in building images, checkerboards, or structured environments.

Output:



Observation:

Red dots appear on corners like window edges, object corners, or tile intersections. Corners are points with high changes in intensity in all directions.

GitHub Link: https://github.com/prajwalawate45/image-processing-algorithms