3D Vision
Exercise 1

Student:

Alireza Amini

1 Task 1

1.1 Introduction

In this section, I evaluate three feature extraction methods, Scale-Invariant Feature Transform (SIFT), Oriented FAST and Rotated BRIEF (ORB), and Harris Corner Detection, along with two feature matching techniques: Brute Force Matching (BF) and Fast Library for Approximate Nearest Neighbors (FLANN). The evaluation focuses on three key aspects:

- The number of candidate feature points detected per frame.
- The number of stable landmarks appearing in at least five consecutive frames.
- The total processing time for each method.

Since Harris only detects keypoints and does not produce descriptors, SIFT descriptors were computed for its detected keypoints to enable evaluation.

1.2 Methodology

1.2.1 Feature Extraction Methods

SIFT: A robust algorithm that detects scale-invariant keypoints and describes them using histograms of gradient orientations. It is known for its high accuracy but computational complexity.

ORB: A more efficient alternative to SIFT, using FAST keypoint detection and BRIEF descriptors. ORB is rotation-invariant and computationally less expensive.

Harris Corner Detector: Detects corners by analyzing intensity changes in local regions. It identifies more corner-like features but was only run on five frames in this experiment.

1.2.2 Feature Matching Methods

Brute Force Matching (BF): Compares every descriptor from one image with all descriptors in another, selecting the closest match based on distance metrics (e.g., Euclidean distance for SIFT and Hamming distance for ORB).

FLANN: A more efficient approach that builds a search tree to approximate nearest neighbors, reducing computational time while maintaining matching accuracy.

1.3 Experimental Results

All evaluations were performed on 200 frames, except for Harris Corner Detection, which was only evaluated on the first twenty frames due to its time-consuming nature. For candidate points per frame, the reported values are computed as averages per frame.

Table 1 presents a comparison of feature extraction methods based on candidate points per frame, while Table 2 compares feature matching techniques in terms of landmark detection and processing time. Note that for Table 1, all feature extraction methods were evaluated using the Brute Force (BF) feature matching algorithm to ensure a fair comparison.

In addition, I present the feature matching results for two consecutive frames using Harris, SIFT, and ORB in the figures 1, 2, and 3.

Table 1: Comparison of Feature Extraction Methods

Method	Candidate Points per Frame	Landmarks	Processing Time (s)
SIFT	2537.16	3309	180.38
ORB	500.00	489	8.95
Harris	4864.68	1047	1711.99

Table 2: Comparison of Feature Matching Methods

Matching Method	Candidate Points per Frame	Landmarks	Processing Time (s)
SIFT + BF	2537.16	3309	180.38
SIFT + FLANN	2537.16	14711	125.94
ORB + BF	500.00	489	8.95
ORB + FLANN	500.00	1738	7.41

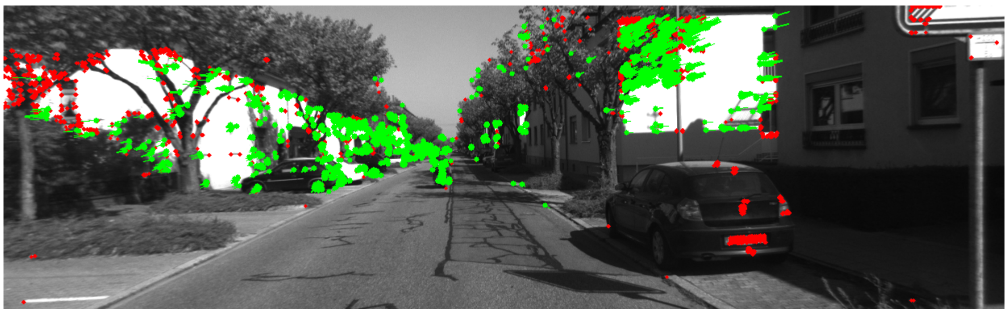

Figure 1: Harris Feature Matching

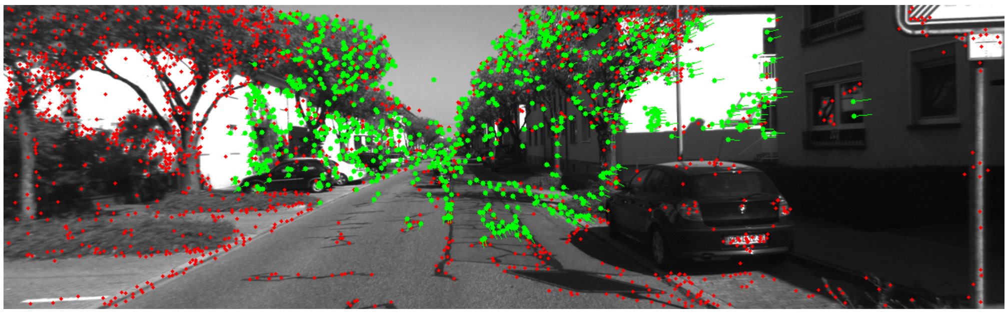
Figure 2: SIFT Feature Matching

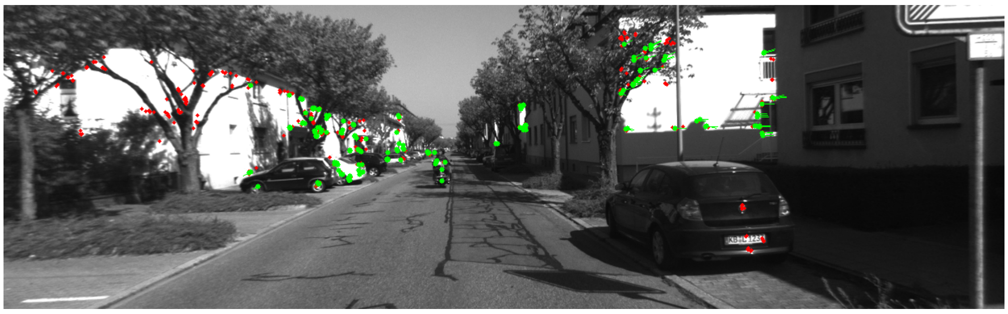
Figure 3: ORB Feature Matching

1.4 Analysis and Discussion

1.4.1 Feature Extraction Comparison

Harris detects the highest number of keypoints per frame (around 4864.68), followed by SIFT (2537.16), while ORB extracts the fewest (500.00). Although Harris identifies a large number of keypoints, it was only evaluated on the first 20 frames, detecting about 1047 stable landmarks. However, its high computational cost makes it impractical for long sequences. SIFT provides robust keypoint detection with a moderate number of features, making it more reliable than ORB, but its execution time is relatively high. On the other hand, ORB is fast and efficient, but the low number of extracted keypoints limits its performance in matching tasks.

1.4.2 Feature Matching Comparison

For feature matching, the combination of SIFT + FLANN achieves the highest number of matched landmarks (14711), showing FLANN's ability to find correspondences efficiently, despite SIFT's high computational cost. In comparison, SIFT + BF results in only 3309 matched landmarks, confirming that FLANN significantly improves performance. For ORB-based methods, ORB + FLANN detects 1738 matched landmarks, much higher than ORB + BF (489 landmarks). Additionally, FLANN reduces the execution time (7.41s vs. 8.95s for BF), making it a better choice overall.

Overall, FLANN outperforms BF in both speed and the number of matched landmarks.

BF struggles to establish stable matches, especially for ORB, whereas FLANN significantly improves results for both ORB and SIFT.

1.5 Conclusion

In summary, SIFT offers robust feature detection with a high number of candidate points but is computationally expensive. ORB, although detecting fewer features, performs efficiently and becomes highly effective when paired with FLANN for matching. Harris, while capable of detecting a large number of candidate points, suffers from extreme computational overhead. Additionally, FLANN-based matching consistently provides superior performance compared to Brute Force Matching, particularly when used with ORB.

2 Task 2

2.1 Introduction

In this task, I explore the effects of a projective transformation on a square and analyze the impact on parallel lines by computing their vanishing points before and after the transformation.

2.2 Methodology

2.2.1 Step 1: Drawing a Square

A square is defined in a 2D coordinate space with the following four corner points:

$$\text{Square Coordinates: } \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

The square is plotted using Matplotlib.

2.2.2 Step 2: Applying a Projective Transformation

A projective transformation matrix H is defined as:

$$H = \begin{bmatrix} 1 & 0.2 & 0 \\ 0.1 & 1 & 0 \\ 0.2 & 0.2 & 1 \end{bmatrix}$$

Each vertex of the square is transformed using H, resulting in a new quadrilateral. The transformed shape is then plotted.

2.2.3 Step 3: Computing Vanishing Points

Vanishing points are computed for parallel lines before and after transformation. For the original shape:

$$\text{Vanishing Point (Original Shape):} \quad (\infty, \infty, 1)$$

For the transformed shape:

$$\text{Vanishing Point (Transformed Shape):} \quad (1, 5, 1)$$

2.3 Results and Visualization

The results are visualized in the following figure:

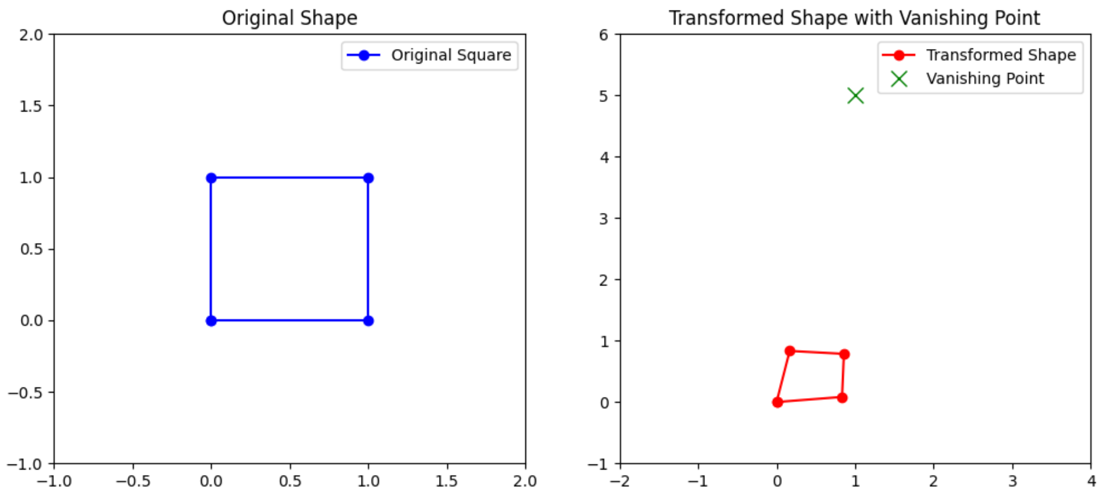

Figure 4: Original and Transformed Shape with Vanishing Points

2.4 Conclusion

This experiment demonstrated how projective transformations affect geometric shapes and parallelism in $\mathbb{R}^2$. The results show that parallel lines in the original shape intersect at infinity, whereas after transformation, they meet at a finite vanishing point.