# Visual Odometry

## Introduction

This report presents an analysis of a monocular visual odometry system implemented in Python using OpenCV. The system employs feature-based methods to extract keypoints, match them across frames, and estimate camera motion through essential matrix decomposition. The implementation follows a classical approach combining ORB (Oriented FAST and Rotated BRIEF) feature detection with FLANN-based matching, followed by essential matrix estimation using RANSAC for robust pose recovery.

The system is designed to process sequences of grayscale images and produce trajectory estimates in real-time, making it suitable for applications requiring continuous localization without external positioning systems. By leveraging the geometric constraints inherent in stereo vision principles applied to temporal sequences, the system can recover both rotation and translation components of camera motion, albeit with an inherent scale ambiguity typical of monocular approaches.

## Key Components

**VisualOdometry Class**: The main processing engine that handles calibration loading, image processing, feature detection, and pose estimation.

**Feature Detection**: Utilizes ORB (Oriented FAST and Rotated BRIEF) detector to identify distinctive keypoints in images, configured to detect up to 3000 features per frame.

**Feature Matching**: Employs FLANN (Fast Library for Approximate Nearest Neighbors) matcher with LSH (Locality Sensitive Hashing) for efficient descriptor matching between consecutive frames.

**Pose Estimation**: Implements essential matrix-based approach using RANSAC for robust estimation, followed by matrix decomposition to recover rotation and translation.

# Implementation

## Calibration System

The system loads camera intrinsic parameters from a NumPy archive file containing the camera matrix. The calibration data is essential for accurate pose estimation and 3D reconstruction. The projection matrix P is computed as K[I|0] where K represents the intrinsic camera parameters.

## Feature Detection and Matching

The ORB detector provides rotation and scale invariant features, making it suitable for visual odometry applications. The FLANN matcher uses LSH indexing optimized for binary descriptors, with parameters configured for 6 hash tables and key size of 12 bits. The system applies Lowe's ratio test with a threshold of 0.8 to filter good matches and reduce false correspondences.

## Essential Matrix Estimation

The system uses cv2.findEssentialMat with RANSAC to robustly estimate the essential matrix from point correspondences. RANSAC parameters include a probability of 0.999 for finding a good model and a pixel threshold of 0.5. The implementation includes validation to check essential matrix rank and singular value properties to detect degenerate configurations.

## Pose Recovery

Essential matrix decomposition yields four possible rotation-translation combinations. The system evaluates each solution by triangulating 3D points and selecting the configuration that maximizes points with positive depth in both camera views. Relative scale estimation uses the ratio of 3D point distances between consecutive poses.

## Trajectory Estimation

The system maintains a cumulative pose matrix, updating it with the inverse of each frame-to-frame transformation. This approach accumulates the camera motion over time, building the complete trajectory in the world coordinate frame.

## Class Methods:

`__init__(self, data_dir, calib_dir)` - Initializes the visual odometry system by loading camera calibration data, images, and setting up feature detection and matching components.

`_load_calib(filepath)` - Loads camera intrinsic parameters from a calibration file containing the camera matrix.

`_load_images(filepath)` - Loads and converts all image files from a directory into grayscale format for processing.

`_form_transf(R, t)` - Creates a 4×4 transformation matrix from a 3×3 rotation matrix and translation vector.

`get_matches(self, i, min_matches=10, visualize=False)` - Detects and matches ORB keypoints between consecutive frames, returning corresponding point pairs.

`get_pose(self, q1, q2)` - Estimates camera pose by computing the essential matrix and decomposing it into rotation and translation.

`decomp_essential_mat(self, E, q1, q2)` - Decomposes the essential matrix into possible rotation-translation pairs and selects the correct solution based on 3D point triangulation.

## Functions:

`visualize_path(path, title, output_file)` - Creates and saves a 2D plot of the estimated camera trajectory in the x-z plane.

`main(data_dir, calib_dir, visualize_matches)` - Main execution function that runs the complete visual odometry pipeline on an image sequence and generates trajectory visualization.

`sum_z_cal_relative_scale(R, t)` - Triangulates 3D points to validate rotation-translation pairs and calculates relative scale between consecutive frames.

---

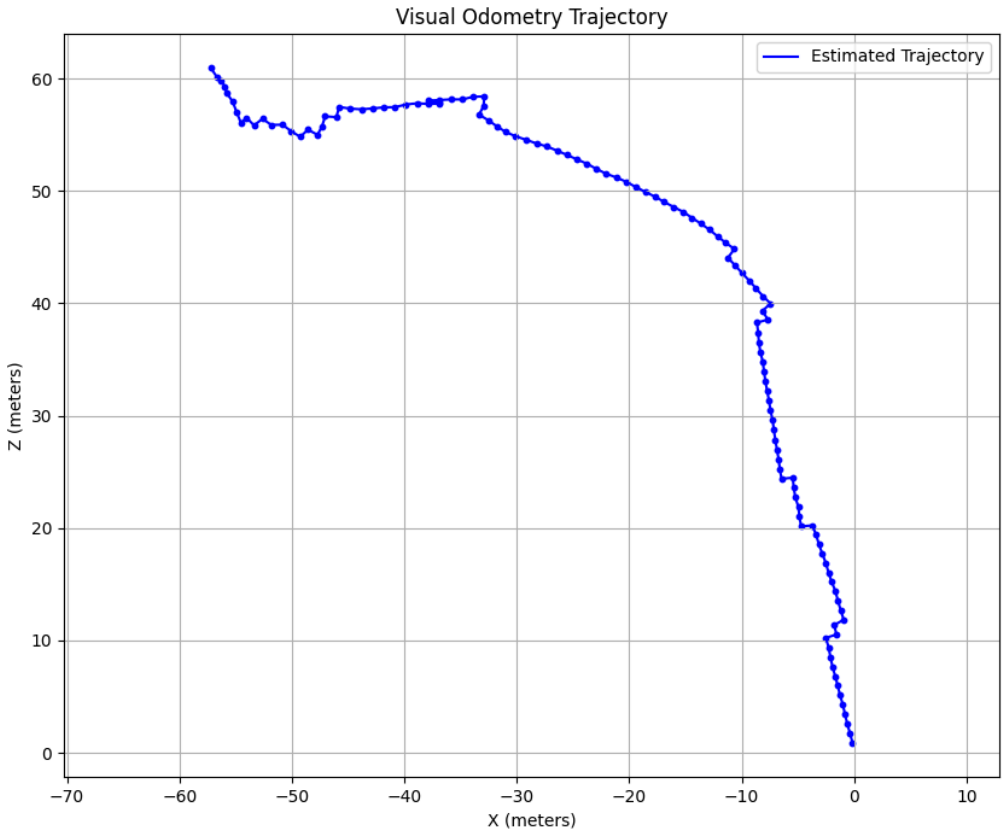
## Error Handling and Robustness

The implementation includes several robustness measures to handle challenging scenarios. Minimum match requirements ensure sufficient correspondences for reliable pose estimation, with a default threshold of 10 matches. The system validates essential matrices by checking singular value

properties and skips frames with insufficient or poor-quality matches. Exception handling covers file loading operations and provides informative error messages for debugging.

## Visualization and Output

The system provides trajectory visualization in the x-z plane, plotting the estimated camera path with connected points. The visualization saves results as PNG images and includes proper axis labeling and grid display. Optional match visualization displays keypoint correspondences between consecutive frames for debugging and analysis purposes.




## Performance Considerations

The ORB detector configuration with 3000 keypoints balances feature density with computational efficiency. FLANN matching provides faster correspondence search compared to brute-force methods, particularly

important for real-time applications. The RANSAC approach offers robustness against outliers but may require parameter tuning for specific datasets.

---

### Team Members:

| Name | ID |
|---|---|
| Youssef Ahmed Kamel | 21011574 |
| Ahmed Mohamed Araby | 21010170 |
| Youssef Mohamed Ahmed | 21011614 |
| Mai Nagah Ali | 21011431 |
| Hassan Fathy Mohamed | 21010463 |

## GitHub

https://github.com/YoussefKamel771/Visual_Odometry.git