Visual Odometry for Autonomous Navigation

Mark Day
Dept of Mechatronics Engineering
University of Canterbury
Christchurch, New Zealand
Msd60@uclive.ac.nz

Prof Richard Green
Dept of Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
richard.green@canterbury.ac.nz

***Abstract*— This paper presents a monocular visual odometry system designed to estimate camera motion using sequential RGB images and corresponding depth information. The method is implemented using a feature-based approach with ORB feature detection and matching, essential matrix estimation, and pose recovery. Depth data is used to resolve the scale ambiguity inherent in monocular systems. The system is evaluated on the TUM RGB-D dataset, and its accuracy is measured by comparing estimated trajectories with ground truth data. The results demonstrate the system's potential for visual navigation in environments relevant to autonomous vehicles, though limitations in rotation estimation and feature matching accuracy are observed.**

Keywords—visual odometry, feature matching, pose estimation, monocular camera, depth scaling, Autonomous Navigation, ORB, Essential Matrix

I. Introduction

Visual Odometry (VO) is a fundamental technique in robotics and computer vision, used to estimate a camera's motion by analysing sequential image frames. Unlike traditional odometry methods that rely on wheel encoders or inertial measurement units (IMUs), VO leverages visual information to compute relative pose, making it especially useful in environments where GPS or other sensors may be unreliable or unavailable [1]. This capability is particularly important in autonomous vehicle navigation, where accurate and robust localization is essential for safe and efficient movement [2]. VO allows a vehicle to perceive motion through its onboard cameras, enabling it to estimate its trajectory in real time, even in unstructured or GPS-denied environments such as indoor spaces, tunnels, or dense urban areas.

Two primary approaches to VO are commonly used: direct methods, which track image intensities to minimize photometric error, and feature-based methods, which extract, match, and track keypoints such as ORB or SWIFT features across frames. Feature-based methods, though computationally more intensive, often provide better robustness in challenging conditions.

In this project, a feature-based monocular VO pipeline is implemented using OpenCV in Python, with the objective of estimating the full trajectory of a moving camera. Since monocular VO lacks inherent scale information, depth data from the TUM RGB-D dataset is integrated to recover absolute motion scale. The method involves detecting features with ORB, matching them using brute-force techniques, and computing the camera pose using the essential matrix. To improve resilience to outliers, a five-point algorithm is used in combination with RANSAC. The resulting pose estimates are scaled using depth information and compared against ground truth trajectories to assess accuracy.

This report presents the theoretical background of the VO method, outlines the implementation steps, and evaluates its performance on benchmark datasets, highlighting its relevance to real-world applications such as autonomous navigation.

II. Background

A. Visual Odometry/Pose

Visual Odometry (VO) is a method of odometry without the use of encoders. Visual odometry uses the change in image frame to frame to estimate pose [3]. There is two main ways of achieving this [4], feature based, and direct VO. Direct VO takes an image sequence and tracks the minimum photometric error between them. This is the simplest way to achieve VO. Feature based VO requires more computation than Direct VO. It takes a sequence of images and applies feature extracting and matching methods to each image such as ORB, SWIFT etc [5]. Then it tracks the minimum reprojection error between the images.

To construct the geometry a 5-point algorithm can be used [6]. This requires five matched points from an image sequence then computes the coefficients for a 10th degree polynomial and root. From this the essential matrix can be formed. If a stereo camera is used a three-point algorithm can be used [7].

B. Essential matrix

The Essential matrix E is defined with the following equation:

$$E = [t]R \tag{1}$$

Where [t] is the skew-symmetric matrix of the translation vector between the camera centres and R is the rotation matrix of the two camera poses [8]. Decomposing the matrix gives the translation and rotation of the camera.

The essential matrix is estimated using a combination of the five-point algorithm [9] and RANSAC [10] to ensure robustness against outliers in feature matching. The five-point algorithm computes the essential matrix from a minimal set of five-point correspondences by solving the epipolar constraint equations, leveraging the fact that the essential matrix has five degrees of freedom. Since real-world data often includes mismatches or noise, RANSAC is used to iteratively sample subsets of point matches, compute candidate solutions, and select the one with the most inlier support [11]. This combination enables reliable estimation of the camera's relative rotation and translation direction between frames.

C. ORB

ORB (Oriented FAST and Rotated BRIEF) is a computationally efficient algorithm that detects and describes image features with scale and rotation invariance [12]. ORB is widely used for visual odometry because it offers a strong balance between speed, accuracy, and computational efficiency.

D. SWIFT

SWIFT (Speeded-Up Wavelet-based Interest Point Feature Transform) is a feature detection and description algorithm designed to provide a balance between robustness and computational efficiency by leveraging wavelet-based analysis [13]. Unlike traditional feature detectors such as ORB, SWIFT employs a multiscale wavelet decomposition of the input image to identify salient points and build descriptors that are inherently resilient to changes in scale, illumination, and moderate affine transformations.

E. Visual SLAM

Visual SLAM can be achieved with both monocular and stereo vision [14]. Monocular Visual SLAM uses a single camera to estimate a device's position and build a map of the environment. It extracts image features (e.g., ORB, SIFT), tracks them across frames, and triangulates 3D points via motion parallax. The camera pose

III. Solution/Method

The proposed method is a using feature based Visual Odometry method with integrating depth image information to achieve scale. This is done in a python program using opencv. The program takes image sequences compares features to match using ORB.

A. Tools

Python – 3.11

Numpy – 1.25.1

Opencv – 4.8.0

Windows – 11

CPU - AMD Ryzen 7 5800H

Data sourced from Computer Vision Group, TUM School of Computation, Information and Technology, Technical University of Munich [15].

B. Method

Both rgb and depth images are loaded into the script using glob. This ensures that they are loaded in, in order. Images are also named using ROS time. This was achieved using a dataset and not real time images.

Firstly, camera intrinsics are used to undistort the image. Camera intrinsics were givin with the data set. This is important when calculating the distance between two matched points.

Two images in sequence have their keypoints and descriptions of those keypoints extracted with ORB feature extraction. Their pixel locations are stored.

Then it is required to identify features that are present in both images as shown in Figure 1. This is made difficult are they will have moved from their original location. This was achieved with a brute force matching technique. For ORB features the normtype must be set hamming as the descriptors are binary.

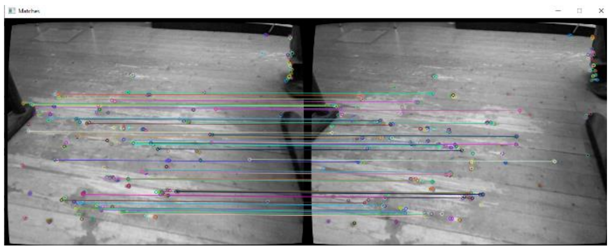

Fig 1. Feature Matching.

If an image had less than 8 matches the sequence was skipped.

An essential matrix was then estimated using the two sets of matched pixel locations and a RANSAC estimation method. This encodes the rotation and translation between the frames.

To extract the rotation and translation of the actual camera the essential matrix is decomposed. Note at this stage the translation of the camera is relative to each frame and not an absolute measurement.

To transform the translation into an absolute measurement it must be scaled. This is achieved by incorporating the depth frames. From the matched 2D points they can be translated into 3D points using the following:

$$X = (u - c_x)\frac{d}{f_x} \quad (2)$$

$$Y = (v - c_y)\frac{d}{f_y} \quad (3)$$

$$Z = d \quad (4)$$

Where (u,v) are match pixel locations and c,f are camera intrinsics. D is the depth value. This is done for both images.

To estimate the absolute scale of motion between two frames using matched 2D keypoints and their corresponding depth maps. It projects the 2D points into 3D space using the camera intrinsics and depth values, filtering out points with invalid depth. For each valid match, the Euclidean distance between the 3D points in the two frames are computed. If too few valid points are found, it defaults to a scale of 1.0 to maintain stability.

The pose is then updated, and this process is repeated for every image sequence.

To evaluate the results against ground truth data from the TUM RGB-D dataset. Both trajectories, associates corresponding timestamps within a specified tolerance, and aligns the VO trajectory to the ground truth using the Umeyama algorithm, which accounts for rotation and translation. Then compute the Absolute Trajectory Error (ATE) as the root mean square error (RMSE) between aligned poses and visualizes both trajectories in a 2D plot for comparison.

IV. Results

A. Dataset rgbd_dataset_freiburg1_floor

This dataset is of an rgb-d camera moving around a set of desks with the camera mainly pointed towards the ground. Figure 2 shows the odometry estimated vs ground truth.

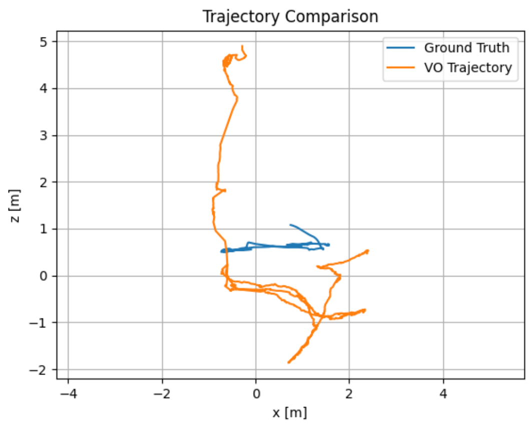

Fig 2. Floor Trajectory and Truth

The Absolute trajectory error, RMSE, is 2.4818m

B. Dataset rgbd_dataset_freiburg1_room

This dataset is shot in a similar environment to the first except the camera is held at facing eye level such that it represents the movement and vision of humans better.

Figure 3 shows the odometry estimated vs ground truth.

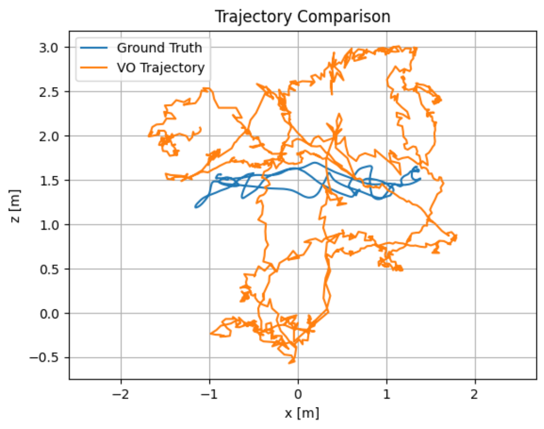

Fig 3. Room Trajectory and Truth

The Absolute trajectory error, RMSE, is 1.3624m

V. Conclusion

A. Conclusion

Results from both datasets are underwhelming. The result from the floor dataset has a total error of 2.4m and observing the graph shows the estimated position of the camera is nowhere the actual. There are features in both trajectories which 'look' similar. It is clear that the program can measure odometry, but This shows that the program can not effectively measure rotation nor translation.

The second dataset shows a similar picture. Its ATE is numerically better than the first dataset 1.3m But inspecting the paths there is far greater noise in the estimated rotation. The two graph show similar results when it comes to translation, but rotation is where the method fails.

C. Comparison to prior results

Prior results in the field using full stereo geometry construct techniques resulted in a maximum error of 5% [7]. Figure 4, shows the resulting map.

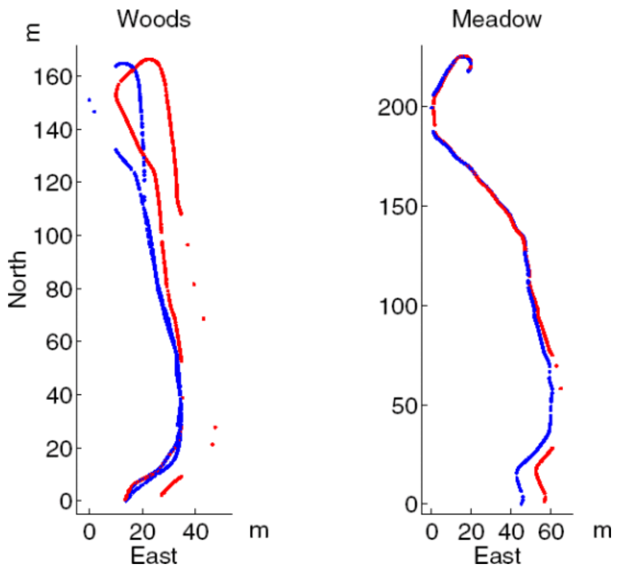

Fig 4. VO from [7]

It shows that the result was better it still struggled to deal with rotation, not to the same degree as the give results.

D. *Limitations*

Two reasons maybe the course of the different results and the nosier trajectory. One the second dataset had had more rotation in it rather than translation hence the better ATE but nosier pathing. Two there was more features that ORB detected, and which would normally add but many had the same descriptions thus misrepresent the scale, translation and rotation.

E. *Further Research*

To remove noise from the trajectory path. Post-processing can be implemented such as standard odometry fliting. Where a low pass filter is implemented on the pose. Different feature techniques could be used such as SIFT or SURF. In this paper only feature matching in 2D was considered but 3D techniques could improve quality.

VI. References

[1] S. A. S. Mohamed, M.-H. Haghbayan, T. Westerlund, J. Heikkonen, H. Tenhunen, and J. Plosila, "A Survey on Odometry for Autonomous Navigation Systems," IEEE Access, vol. 7, pp. 97466–97486, 2019, doi: https://doi.org/10.1109/access.2019.2929133 .

[2] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, Jan. 2006, doi: https://doi.org/10.1002/rob.20103.

[3] Y. Bai, B. Zhang, N. Xu, J. Zhou, J. Shi, and Z. Diao, "Vision-based navigation and guidance for agricultural autonomous vehicles and robots: A review," *Computers and Electronics in Agriculture*, vol. 205, p. 107584, Feb. 2023, doi: https://doi.org/10.1016/j.compag.2022.107584.

[4] S. Kim, I. Kim, L. F. Vecchietti, and D. Har, "Pose Estimation Utilizing a Gated Recurrent Unit Network for Visual Localization," *Applied Sciences*, vol. 10, no. 24, p. 8876, Dec. 2020, doi: https://doi.org/10.3390/app10248876

[5] M.-H. Le, B.-S. Woo, and K.-H. Jo, "A Comparison of SIFT and Harris conner features for correspondence points matching," Feb. 2011, doi: https://doi.org/10.1109/fcv.2011.5739748.

[6] D. Nistér. An Efficient Solution to the Five-Point Relative Pose Problem, IEEE Conference on Computer Vision and Pattern Recognition, Volume 2, pp. 195-202, 2003.

[7] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," *IEEE Xplore*, Jun. 01, 2004. https://ieeexplore.ieee.org/document/1315094

[8] Z. Zhang, "Essential Matrix," *Springer eBooks*, pp. 400–401, Jan. 2021, doi: https://doi.org/10.1007/978-3-030-63416-2_129.

[9] H.-J. Chien, C.-C. Chuang, C.-Y. Chen, and R. Klette, "When to use what feature? SIFT, SURF, ORB, or A-KAZE features for monocular visual odometry," *Image and Vision Computing New Zealand*, Nov. 2016, doi: https://doi.org/10.1109/ivcnz.2016.7804434.

[10] T. Botterill, S. Mills, and R. Green, “Fast RANSAC hypothesis generation for essential matrix estimation.” 2011, https://ir.canterbury.ac.nz/items/9040a188-20ee-46c1-ad08-f6e9b991a571

[11] H. Li and R. Hartley, “Five-Point Motion Estimation Made Easy,” *CiteSeer X (The Pennsylvania State University)*, Jan. 2006, doi: https://doi.org/10.1109/icpr.2006.579

[12] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” *IEEE Xplore*, Nov. 01, 2011. https://ieeexplore.ieee.org/document/6126544

[13] D. Nur and E. Seignez, “Swift Path Planning: Vehicle Localization by Visual Odometry Trajectory Tracking and Mapping,” *Unmanned Systems*, vol. 06, no. 04, pp. 221–230, Aug. 2018, doi: https://doi.org/10.1142/s2301385018500085

[14] R. Mur-Artal and J. D. Tardos, “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras,” *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017, doi: https://doi.org/10.1109/tro.2017.2705103.

[15] “Computer Vision Group - Dataset Download,” *cvg.cit.tum.de*. https://cvg.cit.tum.de/data/datasets/rgbd-dataset/download