COMP0130: Robot Vision and Navigation - Group K
Coursework 3: Monocular ORB-SLAM2

Hyungjoo Kim	**Ahmed Salem**	**Zhonghao Wang**
170103694	20208009	20061720
ucabhki@ucl.ac.uk	zcemasa@ucl.ac.uk	ucabzw6@ucl.ac.uk

1 Introduction

One of the key challenges of autonomous navigation of an unknown environment is to be able to self-localise in an environment that the robot has not encountered previously (Silveira, Malis and Rives, 2008). This concept is known as simultaneous localisation and mapping (SLAM) and has been explored in previous assignments of the COMP0130 module using vehicle odometry as the primary tool for vehicle pose estimation. SLAM has also been implemented using a computer vision approach in the last decades particularly attracting companies interested in augmented reality (AR) (Klein and Murray, 2007; Mur-Artal and Tardos, 2017a). Visual SLAM approaches require a robust, accurate and automatic estimation of the 3D motion of the camera's pose (mounted on the robot) sequentially using images from the next frame (Klein and Murray, 2007; Mur-Artal and Tardos, 2017a). The images are used in a technique known as *extensible tracking* to extract static elements from the environment and add to an initial map contributing to the "Mapping" term of the SLAM acronym (Klein and Murray, 2007). This is made possible as typically 2 frames are quite close together in time and space assuming a standard frame rate of 30 Hz. To put simply, Visual SLAM examines the changes that the motion induces on the images captured to incrementally estimate pose of the vehicle while simultaneously producing an estimated globally consistent map (Agapito, 2021).

Klein & Murray produced one of the most important papers in visual SLAM techniques in their 2007 paper entitled *Parallel Tracking and Mapping for Small AR Workspace* where they concluded that the tracking and mapping did not necessarily need to occur in real-time (Klein and Murray, 2007). They divided vision-based SLAM into two threads, a tracking thread and a mapping thread. This enabled AR applications to occur in real-time without the use of GPUs for acceleration (Mur-Artal, Montiel and Tardos, 2015). From a high level perspective, the tracking thread does occur in real-time and is responsible for estimating the camera pose robustly (Agapito, 2021). It mostly follows on the work of Mouragnon *et al.* of visual odometry. The mapping thread occurs at user-specified keyframes (not in real-time) and is responsible for creating rich, dense and accurate maps (Agapito, 2021).

The aim of this paper is to use and evaluate ORB-SLAM2, a modern visual SLAM system, under different conditions using the simplest setting, monocular mode. This will include a detailed description of the tracking thread and a comprehsive evaluation of each of the four different conditions assessed. For each condition, the evaluation will follow the approach of Sturm *et al.* and the results will be discussed (Sturm *et al.*, 2012).

2 Tracking Thread

As mentioned in the introduction, the steps of the tracking thread is performed for each frame of the camera and its frequency is subsequently matched to the frame rate of the camera, typically 30Hz for the monocular setting (Mur-Artal, Montiel and Tardos, 2015). The tracking thread consists of four high level steps which are detailed below: ORB extraction, initial pose estimation from previous frame, track local map and new keyframe decision as shown in **Figure 1**.

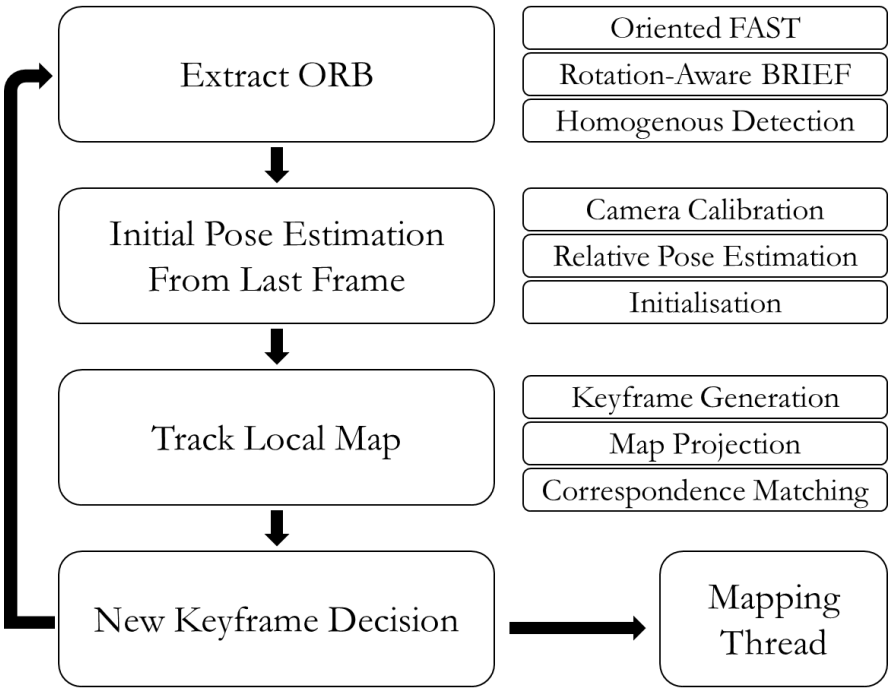

Figure 1: Steps of the tracking thread with details of the components and constituents for ORB-SLAM2.

2.1 Find Features/Extract ORB

For feature extraction, ORB-SLAM2 uses the ORB (Oriented FAST and Rotation-Aware BRIEF) algorithm which builds on the work of Rosten & Drummond and Calonder *et al.* for FAST and rBRIEF respectively (Rublee *et al.*, 2011). FAST (Features from Accelerated Segment Test) classifies features as "corners" based on a set of initial conditions and is accelerated by machine learning (ML) decision trees to improve efficiency. Non maximal-suppression is then applied to remove less-adhering corners and to improve the system's overall performance (Rosten and Drummond, 2006). FAST however does not describe the orientation of a corner (Rublee *et al.*, 2011).

rBRIEF is a feature descriptor that is efficient in computational speed and memory efficiency by using binary tests between pixels to match/recognise features (Rublee *et al.*, 2011). It is however, very sensitive to rotation as it operates on the notion of high variance for discrimination which is lost when attempting to steer rBRIEF according to the orientation of the features (Rublee *et al.*, 2011). Rublee *et al.* produced rBRIEF which uses a greedy ML algorithm to recognise features that are uncorrelated and thus have higher variance (Rublee *et al.*, 2011).

ORB-SLAM2 extracts corners and applies feature descriptors using ORB homogenously by dividing each image into cells depending on resolution. The algorithm aims to extract an identical number of corners for each cell but does not apply it for textureless cells using an outlier detection component (Mur-Artal, Montiel and Tardos, 2015).

2.2 Camera Pose Estimation/Update from last Frame

2.2.1 Camera Calibration

In the monocular camera mode, ORB-SLAM2 uses the laws of similar triangles (based on the pinhole camera model) to determine the odometry of the robot visually as shown in **Figure 2** (Mur-Artal and Tardos, 2017b). Typically for visual SLAM in the monocular mode, the camera focal length is known. The pinhole camera model transforms 3D points $(\mathbf{X}_c)$ in the camera reference (denoted by the subscript $\mathbf{c}$) into 2D points $(\mathbf{x}_c)$ on the image plane.

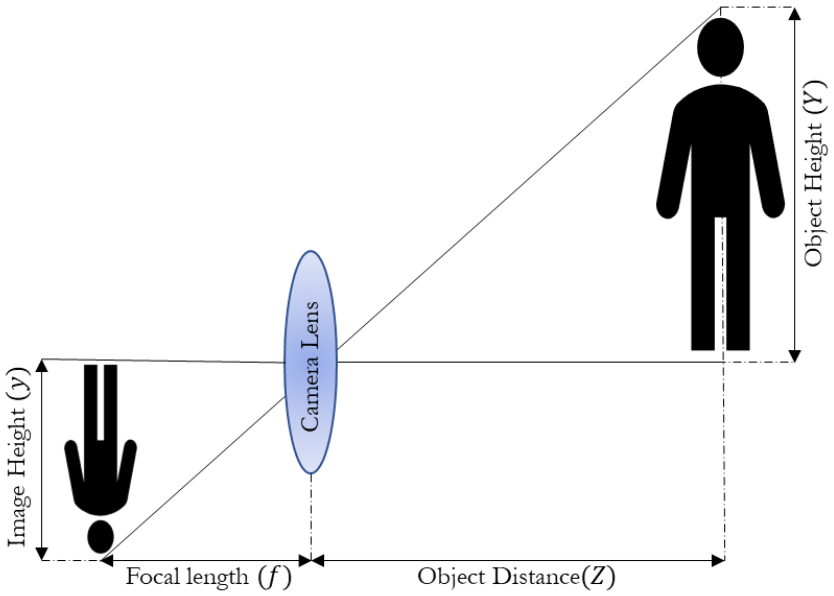

Figure 2: Mathematics of perspective image showing the pinhole camera model which assumes that laws of similar triangles can be used to predict objects in the 3D space from an image. Image recreated from (Agapito, 2021).

The laws of similar triangles are given by equation (1),

$$\frac{y}{f} = \frac{Y}{Z} \tag{1}$$

where y is the image height, f is the camera's focal length, Z is the object distance, and Y is the object's height as shown in **Figure 2**. This causes the relationship between the 3D and 2D frame to be given by equation (2) where in the 2D frame the objects have coordinates in the image plane defined by x_c and y_c (Agapito, 2021).

$$\mathbf{x}_c = \lambda \mathbf{X}_c, \text{ where } \lambda = \frac{f}{Z_c}, \quad \mathbf{x}_c = \begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix}, \text{ and } \mathbf{X}_c = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{2}$$

This can be rewritten to a homogenous coordinate system by incorporating a 3×4 projection matrix which represents a map from 3D to 2D as shown in equation (3) (Agapito, 2021).

$$\begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{3}$$

Images captured by the monocular camera are discretised into pixels causing a need for the camera coordinate system to be converted to the image coordinate system $(\mathbf{x}_i)$. In $\mathbf{x}_i$, the origin is the upper left corner and thus there is an offset between both coordinate systems that must be calibrated for as shown in **Figure 3** (Agapito, 2021). If k_u and k_v are defined as number of pixels per length of image in the u and v directions as shown in **Figure 3**, then equation (4) can be used to define the translation between the coordinate systems, where u_0 and v_0 are the principal point's coordinates (Agapito, 2021).

$$k_u x_c = u - u_0, \quad k_v y_c = v_0 - v \tag{4}$$

Subsequently, the relation between $\mathbf{x}_i$ and $\mathbf{x}_c$ can be derived as shown in equation (5), where K is the camera calibration matrix.

$$\mathbf{x}_i = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & -fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}_c = \mathrm{K}\mathbf{x}_c \tag{5}$$

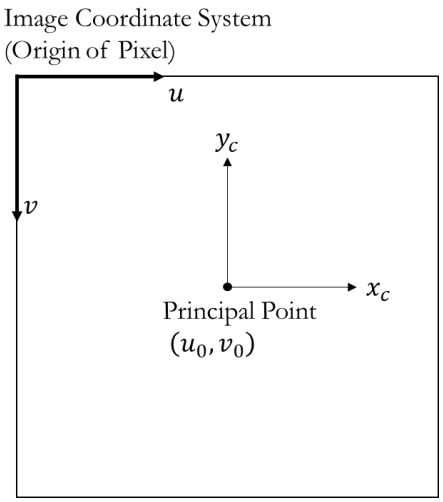

Figure 3: A square image is shown where the origin of the image coordinate system is shown as being in the upper left corner and the principal point is shown as centre of the image where the optic axis intersects the image plane. This is accounted for when transforming coordinate systems using a camera calibration matrix. Image recreated from (Agapito, 2021).

Finally, as the camera is moving, it is necessary to express objects in terms of the world coordinate system ($\mathbf{X}_w$) in terms of $\mathbf{X}_c$. This can be related using a Euclidean transformation as defined by equation (6), where R is a 3×3 rotation matrix and $\mathbf{t}$ is a 3×1 translation vector used to define the rotation and translation from $\mathbf{X}_w$ to $\mathbf{X}_c$ (Agapito, 2021).

$$\mathbf{X}_c = \mathrm{R}\mathbf{X}_w + \mathbf{t} \quad \text{or} \quad \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{R} & \mathrm{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{6}$$

Thus by substituting and rearranging the relations defined by equations (2), (3), (5) and (6), $\mathbf{x}_i$ can be related to the 3-D space as shown by equation (7) (Agapito, 2021). This may also be rearranged to solve for the pose of the robot in $\mathbf{X}_w$.

$$\mathbf{x}_i = \mathrm{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \mathrm{K}[\mathrm{R}|\mathbf{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{7}$$

2.2.2 Relative Pose Estimation

To predict the vehicle odometry it is necessary to estimating the relative pose between two consecutive frames. Post intrinsic camera calibration, it is possible to assume a spherical model for the camera. This model is used to map 3D points into the 2D frame. Between 2 consecutive frames, the ORB-SLAM2 system initially identifies corresponding corners/features using ORB as mentioned in section 2.1 (Rublee *et al.*, 2011; Agapito, 2021). If p_1 and p_2 are defined as vectors containing image coordinates pointing from the camera between 2 frames towards a corresponding feature point as shown in **Figure 4** and in equation (8), an epipolar plane can be formed. Using the laws of coplanarity, a 3×3 essential matrix (E) can be

derived which describes the corresponding points in the 2 images using the translation (**t**) and rotation (R) matrices as shown in equations (9)-(11) (Agapito, 2021).

$$p_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}, p_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} \tag{8}$$

$$p_2^T \cdot (t \times p_1') = 0 \quad \text{where} \quad p_1' = R p_1 \tag{9}$$

$$p_2^T \, E \, p_1 = 0 \quad \text{where} \quad E = [t]_{\mathrm{x}} \, R \tag{10}$$

$$[t]_{\mathrm{x}} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{11}$$

The relations in equations (9)-(11) are a direct result of the epipolar constraint described in equation (10). This method of calculating the essential matrix is known as singular value decomposition (SVD) where, as explained, the matrix is only dependent on rotation and translation of the corresponding features between the two images.

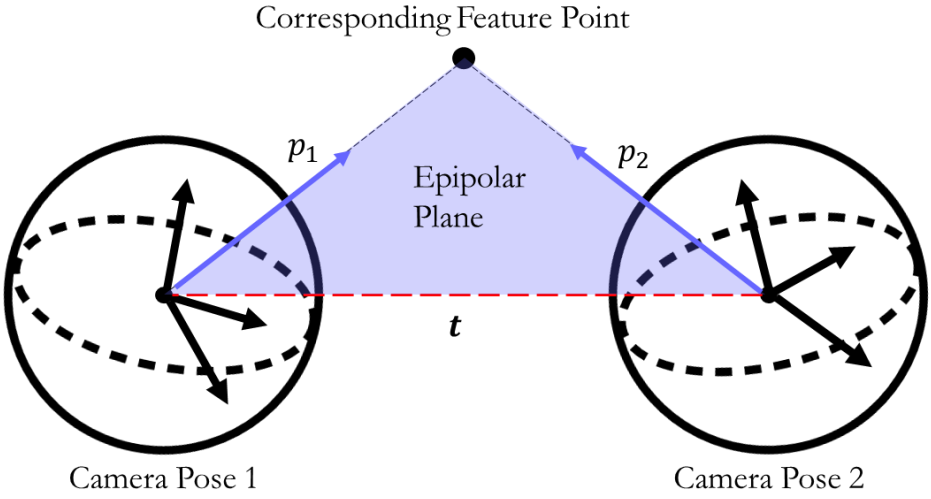

Figure 4: A graphic showing the epipolar plane formation between a camera pose in 2 consecutive frames to outline how the essential matrix is calculated. Image recreated from (Agapito, 2021).

The essential matrix, if estimated robustly and accurately, can be used to find corresponding points in consecutive frames as the points can only lie along the epipolar lines on the new image. This turns matching corresponding points into a 1D problem which allows for more correspondences to be found enhancing the performance of the pose estimation.

2.2.3 Initialisation

In the monocular mode, ORB-SLAM2 assumes a constant velocity motion model to estimate two different geometrical models in parallel to account for two scene types, non-planar and planar (Mur-Artal, Montiel and Tardos, 2015). The homography matrix for the planar scene is shown in equation (12), where the subscript "c" refers to the current frame, and the subscript "r" refers to the reference frame.

$$\mathbf{x_c} = \mathbf{H_{cr}} \, \mathbf{x_r} \tag{12}$$

The essential matrix described in section 2.2.2 is computed from the fundamental matrix (**F**). **F** is used when a camera is not calibrated or for the stereo mode setting on ORB-SLAM2 (Agapito, 2021). For the

non-planar scene geometrical model, the fundamental matrix is computed using equation (13) (Mur-Artal, Montiel and Tardos, 2015).

$$\mathbf{x}_c^T \, \mathbf{F}_{cr} \, \mathbf{x}_r = 0 \tag{13}$$

To convert back to the essential matrix, equation (14) can be used where, as in section 2.2.1, K refers to the camera calibration matrix (Mur-Artal, Montiel and Tardos, 2015).

$$E_{rc} = \mathrm{K}^T \, \mathbf{F}_{rc} \, \mathrm{K} \tag{14}$$

Then the algorithm performs as detailed in the steps below (Mur-Artal, Montiel and Tardos, 2015):

1. Find initial correspondences in the current and reference frames. A threshold is defined where if not enough matches are found, the reference frame is reset.
2. Compute the homography and fundamental matrices using RANSAC. RANSAC is a robust algorithm that estimates the models using the approach detailed below. It aims to predict a model and minimise the error.
    a. Randomly select points to determine model parameters (8 for fundamental, 4 for homography)
    b. Solve for model parameters and calculate the error for all data points outside of estimate and predefined tolerance.
    c. If the ratio of inliers to outliers exceeds a set threshold, re-estimate the model parameters and terminate.
    d. Else, repeat for the maximum number of iterations (8 for fundamental, 4 for homography) to minimise the estimated model error.
3. Compute scores (S_M) for each model (M) to decide which model should be used. Equation (15) details the scoring as explained by (Mur-Artal, Montiel and Tardos, 2015).

$$\begin{gathered} S_M = \sum_i \left(\rho_M\left(d_{cr}^2(\mathbf{x}_c^i, \mathbf{x}_r^i, M)\right) + \rho_M\left(d_{rc}^2(\mathbf{x}_c^i, \mathbf{x}_r^i, M)\right)\right) \\ \rho_M(d^2) = \begin{cases} \Gamma - d^2, & \text{if } d^2 < T_M \\ 0, & \text{if } d^2 \geq T_M \end{cases} \end{gathered} \tag{15}$$

    ρ_M represents the Spearman's rank coefficient for scoring, d is the symmetric transfer error between the frames, T_M is the pre-set threshold for outlier rejection, Γ is the gamma function defined to equal the homography matrix threshold, T_H, to ensure that models score equally (Mur-Artal, Montiel and Tardos, 2015).
4. Select model for camera pose estimation using the heuristic (R_H) defined in equation (16).

$$R_H = \frac{S_H}{S_H + S_F} \tag{16}$$

    Empirically, (Mur-Artal, Montiel and Tardos, 2015) have found that if $R_H > 0.45$, then the homography matrix should be used. Otherwise, the fundamental matrix is used.

2.3 Track Local Map (Draw Graphics)

ORB-SLAM2 is a keyframe-based SLAM solution, which relies heavily on optimization to estimate both motion and structure. The keyframe-based optimisation takes place at defined intervals in the sequence (Younes *et al.*, 2017). ORB-SLAM2 localises in the world coordinate system by using the map points (correspondences) generated as described in the previous section. Each map point (p_i) stores a 3-D position, a viewing direction, an ORB descriptor, and maximum and minimum distances at which a point can be observed to detail a scale invariance region (Mur-Artal, Montiel and Tardos, 2015). Keyframes store the pose, the camera intrinsics, and the features extracted from the frame to localise and relocalise in a sequence as described earlier (Mur-Artal, Montiel and Tardos, 2015).

Following an estimation of the pose and the identification of correspondences (map points) from the previous frame, the ORB-SLAM2 algorithm projects the local map intro the frame to search for more map correspondences (Mur-Artal, Montiel and Tardos, 2015). A local map is projected rather than a global map due to the complexity of larger global maps.

As the keyframes are generated at intervals, and not in real time, the local map is projected using sets of keyframes (κ). 3 keyframe types are present, keyframes which have correspondences in the present frame, (κ_1), (κ_2) which has the neighbours of κ_1 in the covisibility graph, and a reference keyframe (κ_{ref}) which shares the majority of correspondences with κ_1 (Mur-Artal, Montiel and Tardos, 2015). All map points are then searched in the current frame as described by Mur-Artal, Montiel and Tardos. Below is the paraphrased steps from (Mur-Artal, Montiel and Tardos, 2015) for how the algorithm behaves for each map point.

1. Generate the map projection for the current frame. If a map point is out of the image bounds, reject it.
2. Compute angle between map point's direction and the camera's viewing direction. If the angle is larger than 60°, reject the map point.
3. Compute the distance (d) to the map point using the estimated camera pose. If the map point is not in the scale invariance region, reject the point.
4. Compute the scale by using the distance ratio, d/d_{min}, where d_{min} is the minimum distance of the scale invariance region.
5. Use the ORB descriptor to compare the observed map point with the unmatched features in the frame. Associate the map point to the most matching map points.

Once steps 1-5 are completed, the algorithm optimises with all the map points in the frame to effectively localise in the world coordinate system.

2.4 New Keyframe Decision

Keyframes are only used if the ORB-SLAM2 algorithm determines that the tracking is successful according to the below criteria detailed by (Mur-Artal, Montiel and Tardos, 2015).

- >20 frames have passed since last keyframe insertion.
- >20 frames have passed since last relocalisation.
- At least 50 map points are present in the current frame.
- Current frame tracks less than 90% of κ_{ref}.

These conditions enable successful tracking particularly in challenging environments or trajectories as it generates keyframes quite frequently. Finally, the generated keyframes are then used by the mapping thread of ORB-SLAM2 as mentioned in the introduction.

2.4.1 Motion-only Bundle Adjustment

The algorithm uses bundle adjustment (BA) to optimise the camera pose using the correspondences between frames in the world coordinate system. Many forms of BA are present in other threads of ORB-SLAM2, but only pose-optimisation BA is discussed here as it is part of the tracking thread (Mur-Artal and Tardos, 2017a). This method of BA makes an assumption of a constant velocity model and is used to reject outlier observations (Mur-Artal and Tardos, 2017a). This is performed by minimising the reprojection error of the matched 3D correspondences to the keypoints (x^i) using a Huber cost function (ρ_h) as shown by equation(17) (17) and (7) (Mur-Artal, Montiel and Tardos, 2015; Mur-Artal and Tardos, 2017a).

$$\{\mathrm{R}, \mathbf{t}\} = \underset{\mathrm{R}, \mathbf{t}}{\operatorname{argmin}} \sum_{i \in \chi} \rho_h \left(\left\| \mathrm{x}^i - \pi(R\mathbf{X}_w + \mathbf{t}) \right\|_{\Sigma}^2 \right) \tag{17}$$

$$\pi\left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}\right) = \begin{bmatrix} f_x \dfrac{X}{Z} + c_x \\ f_y \dfrac{Y}{Z} + c_y \end{bmatrix} \tag{18}$$

π refers to the projection functions of features in a keypoint, where unlike in equation (1), the focal lengths (f_x, f_y) are not assumed to be identical in the x and y directions respectively. c_x and c_y is the principal point, in the x and y directions (similarly to u_0 and v_0 in Equation (4)). $X, Y,$ and Z are the locations of the map points/features in $\mathbf{X}_w$. R and $\mathbf{t}$ have the same meanings as in previous sections. Σ is the covariance matrix associated with the scale where the keypoint lies relative to the camera.

ORB-SLAM2 stores matched map points to keyframes using a motion from structure 3-D to 2-D image correspondences, where each point is transformed from the world coordinate system to the image coordinate system (Mur-Artal and Tardos, 2017a; Agapito, 2021). In equation (17), $\mathrm{x}^i - \pi(R\mathbf{X}_w + \mathbf{t})$ refers to the error term which links the map point with the keyframe-stored point. By minimizing this reprojection error term with ρ_h, the camera pose is optimised.

This configuration of BA minimises the computational energy required as pose optimisation in the tracking thread only occurs following keyframes usually generated at 1 Hz, which is less frequent than a standard camera frame rate of 30 Hz.

3 Evaluation

The TUM dataset and the KITTI dataset are presented in the papers (Sturm *et al.*, 2012) and (Geiger, Lenz and Urtasun, 2012), respectively. Sturm *et al.* used two different forms of error metrics for a qualitative and quantitative analysis of camera trajectory estimation. The two different forms of error metrics consist of absolute trajectory error (ATE) and relative pose error (RPE). The absolute trajectory error refers to the absolute performance error of visual odometry and to the absolute error of its actual position regardless of time. In other words, it is an easy way to evaluate global consistency by comparing the absolute distances between the ground truth trajectory and the estimated camera trajectory. The paper (Horn, Hilden and Negahdaripour, 1988) presented a closed-form solution for the least-squares solution of absolute direction that aligns the ground truth trajectory and the estimated camera trajectory to establish an arbitrary coordinate frame. A rigid-body transformation (S) can be calculated by the sequence of camera poses (n) from the ground truth trajectory ($Q_1, \dots, Q_n \in \mathrm{SE}(3)$) and the estimated camera trajectory ($P_1, \dots, P_n \in \mathrm{SE}(3)$) using the least-squares solution. After computing the transformation, ATE at the time step (t) can be calculated as follows:

$$F_t \coloneqq {Q_t}^{-1} S P_t \tag{19}$$

In order to evaluate the ATE between the trajectories, the root mean squared error (RMSE) over all time with the translational components of ATE (F_t) is suggested (Sturm *et al.*, 2012).

$$\mathrm{RMSE}(F_{1:n}) \coloneqq \sqrt{\frac{1}{n}\sum_{t=1}^{n} \|trans(F_t)\|^2} \tag{20}$$

Although the ATE can also be evaluated by adding the rotational component, Sturm *et al.* recommended evaluating only translational error because the rotational error is shown as the translational error when the camera moves.
On the other hand, the relative pose error (RPE) refers to the evaluation of the visual odometry system and how much error occurs over time, which corresponds to the drift of the trajectory. The RPE can be calculated using a similar computation method, but it needs to consider a fixed time interval (Δ). The RPE (E_t) can be represented as follows:

$$E_t := \left(Q_t{}^{-1} Q_{t+\Delta}\right)^{-1} \left(P_t{}^{-1} P_{t+\Delta}\right) \tag{21}$$

Similar to evaluate the RPE (E_t), RMSE over all time can be computed with the translational component

$$\mathrm{RMSE}(E_{1:n}, \Delta) := \sqrt{\frac{1}{m}\sum_{t=1}^{m} \|trans(E_t)\|^2} \tag{22}$$

, where m is the individual RPE along the sequence of poses (n), and it can be defined as $m = n - \Delta$.

The monocular ORB SLAM system typically uses 30 frames per second at the steps of the tracking thread as mentioned in section 2, so it uses more than one previous frame to estimate the pose of the camera for every single image. The delta can be set to 30, which provides the drift per second in a sequence. In other words, between the start frame and the end frame can be compared directly, which can be defined as $\Delta = n$. Therefore, the RPE based on the tracking tread for the monocular ORB SLAM system suggests computing the average over all possible time intervals.

$$\mathrm{RMSE}(E_{1:n}) := \frac{1}{n}\sum_{t=1}^{n} \mathrm{RMSE}(E_{1:n}, \Delta) \tag{23}$$

Furthermore, the ATE from the above method, sum of square error (SSE), and standard deviation (SD) will be evaluated in this paper, and its camera trajectory estimation can be easily evaluated using the EVO tool, where more information is provided at http://github.com/MichaelGrupp/evo.
The SSE can be calculated using the mean of the translational components of ATE (F_t) during the sequence of poses. It can be represented as follows,

$$\mathrm{SSE}(F_{1:n}) := \sum_{t=1}^{n} (trans(F_t) - \overline{trans(F_t)})^2 \tag{24}$$

where $\overline{trans(F_t)}$ is the mean value of the translational components of ATE, and n is the number of sequences.

Similarly, to calculate the SD, equation (25) can be used,

$$\mathrm{SD}(F_{1:n}) := \sqrt{\frac{\sum_{t=1}^{n} (trans(F_t) - \overline{trans(F_t)})^2}{n}} \tag{25}$$

where $\overline{trans(F_t)}$ is the mean value of the translational components of ATE, and n is the number of sequences.

Therefore, the EVO tool provides the estimation of camera trajectory with several metrics, which are the absolute trajectory error (ATE), the sum of square error (SSE), and the standard deviation (SD), respectively. However, the 7 degrees of freedom alignment must be satisfied to use the EVO tool. In the following sections, details of the alignment will be explained, and the above metrics will be evaluated depending on four different conditions, which evaluate off-the-shelf options, control the number of ORB features, turn off the outlier rejection, and turn off the loop closure, respectively.

3.1 Default Settings Evaluation

This section evaluates the off-the-shelf option, which is the default setting. The estimated camera trajectory from the monocular ORB-SLAM2 is evaluated using the KITTI sequence. As mentioned above, the 7 degrees of freedom (DoF) must be satisfied to evaluate the results using the EVO tool because the monocular system requires the 7-DoF alignment, which comprises of 3 rotational, 3 translational, and one scale component. The KITTI ground truth data needs to align timestamps and camera poses (Agapito, Wang and Lu, 2021). This data should be converted to one that builds the alignment using mono_kitti.cc file so that the TUM format of the ground truth can be provided. After that, the estimated camera trajectory from the KITTI sequence and the ground truth trajectory from the TUM format can be evaluated using the metrics in the EVO tool.

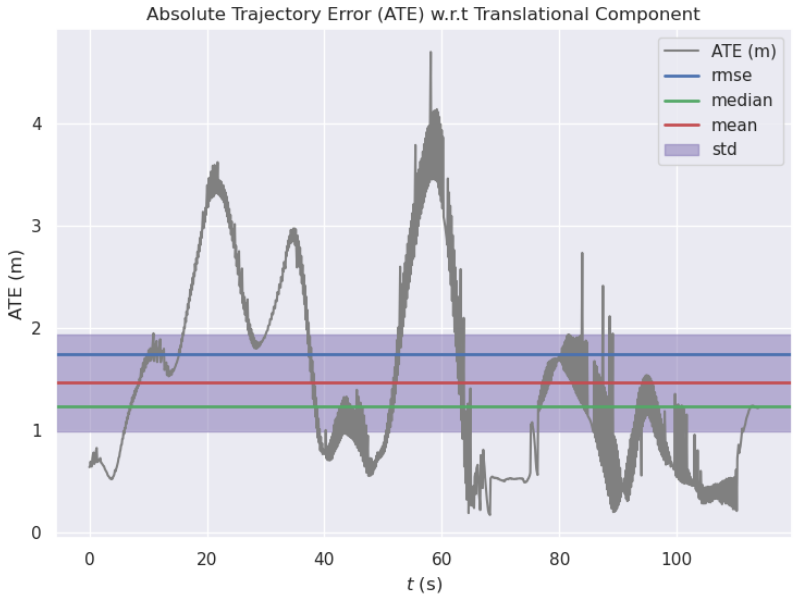

Figure 5: Evaluation of the absolute trajectory error between the ground truth trajectory and the estimated camera trajectory with the 2000 number of features using the 07 KITTI sequence.

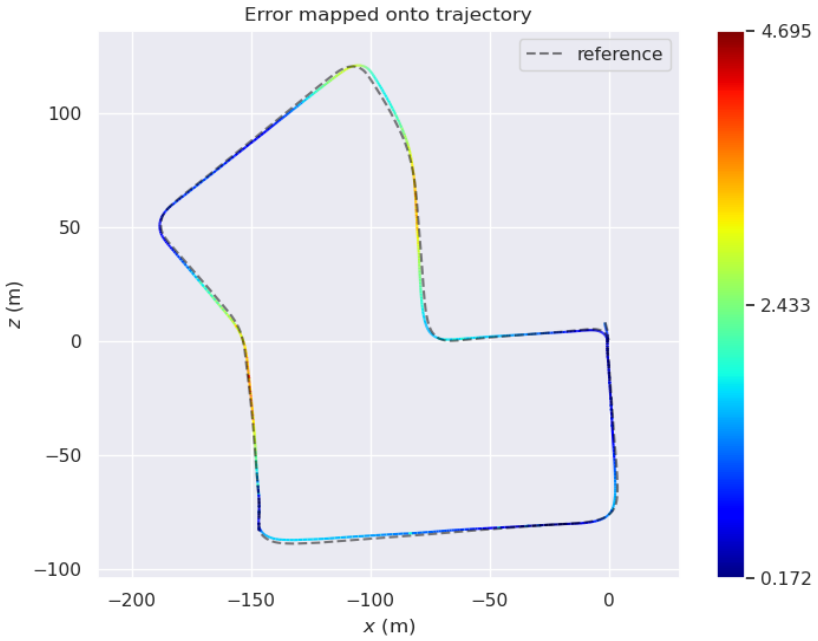

Figure 6: Evaluation of the error mapping between the ground truth trajectory and the estimated camera trajectory with the 2000 number of features using the 07 KITTI sequence.

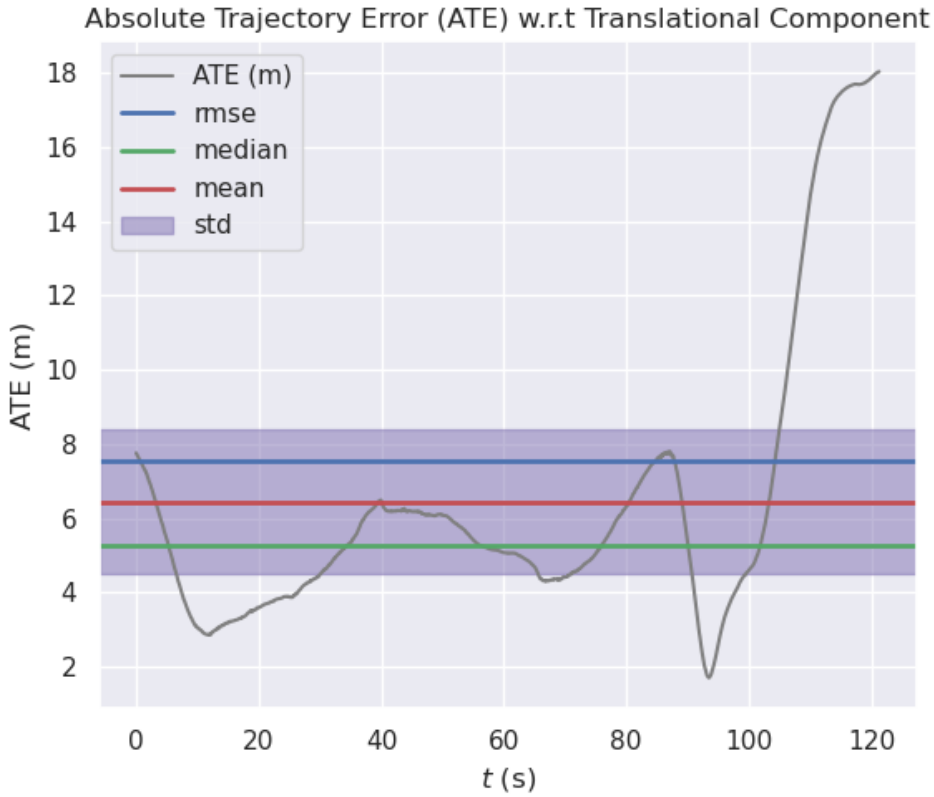

Figure 7: Evaluation of the absolute trajectory error between the ground truth trajectory and the estimated camera trajectory with the 2000 number of features using the 10 KITTI sequence.

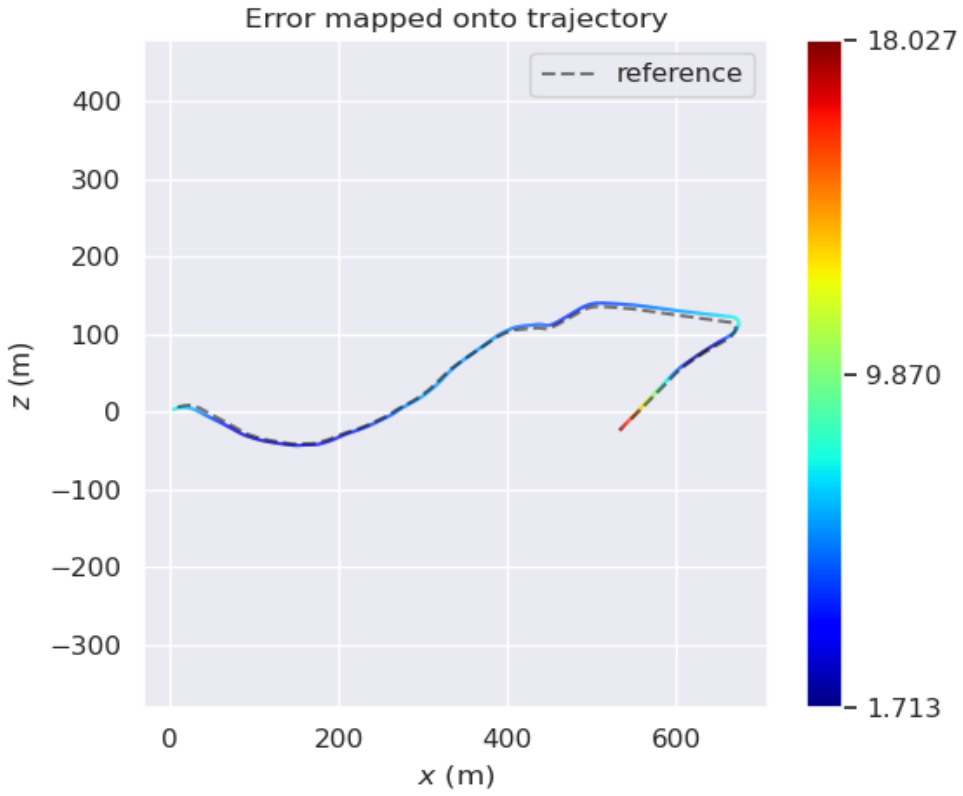

Figure 8: Evaluation of the error mapping between the ground truth trajectory and the estimated camera trajectory with the 2000 number of features using the 10 KITTI sequence.

Figure 5 and **Figure 6** represent the values of metrics and error mapped onto the trajectories between the ground truth and the estimated camera using the 07 KITTI sequence. **Figure 7** and **Figure 8** also show the values of the metrics, and the error mapped onto the trajectories of the 10 KITTI sequence. This experiment uses KITTI04-12.yaml, which provides, as the off-the-shelf options, the number of ORB features to be set to 2,000. In other words, this experiment is conducted by extracting the features properly, detecting the loop closure, and fully optimising the process to provide the results.

Table 1: Evaluation of the metrics of the ATE on the off-the-shelf options with the 07 and the 10 KITTI sequences.

Value (m)	KITTI sequence	
	07	10
RMSE of ATE	1.741537	7.537200
SSE of ATE	3324.116	66466.98
SD of ATE	0.942054	3.900002
Maximum of ATE	4.695008	18.02683
Minimum of ATE	0.171636	1.712913
Mean of ATE	1.464748	6.449758
Median of ATE	1.233301	5.270552

The monocular system inherently requires consecutive frames to perform 3-dimensional reconstruction using a single camera that makes it difficult to obtain 3D points. Furthermore, the system results in a scale variance problem between the frames, which cannot provide an accurate translation. However, the matching feature pairs in two recent frames can be initialised to use the obtained transformation and make them into 3D points using triangulation during the initialisation to provide the results based on the principles from section 2. In addition, the appropriate number of features, outlier rejection, and loop closure are applied to this experiment that guarantees better results.

The 07 KITTI sequence shows navigation through a suburban environment with houses, cars, and small streets. **Figure 5** shows the ATE with respect to the translational component, and the mean value of ATE from **Table 1** is relatively small, which means that the errors are small. **Figure 6** shows the error between the ground truth and the estimated camera trajectory that can be intuitively compared. For this analysis, drift paths are defined as the point of paths with significant rotational components. It can be seen that the parts of the straight road are significantly similar, on the other hand, the parts of the drift path slightly deviate from the exact road. **Figure 5** shows that the ATE with the translational component significantly fluctuated to increase five-fold, whereas the drift path (approximately 90 degrees) is present 5 times in the loop. Furthermore, the trajectory contains one loop, and the ATE from **Figure 5** represents the error decreases at the end of the sequence, which proves that loop closure is important for the performance in the monocular system because the system detects matching between the extracted feature points from the early stage and the corresponding the feature points from the end stage.

On the other hand, the 10 KITTI sequence is also the same environment, but the error values from Table 1 shows that it is bigger than the 07 KITTI sequence. This is because the 07 sequence has many straight roads and a few drift roads, but the 10 sequence has many winding roads, as shown in **Figure 8** so that the rotational and translational camera poses are continuously changed. **Figure 7** shows that the ATE with respect to the translational component vastly increases at the end of the sequence, where the part of the big drift (approximately 140 degrees) is present. Furthermore, the ATE with respect to the translational component increases with the distance.

The data shows that drift paths increase ATE, and that loop closure increases the performance when the sequence has a loop as shown in the above figures. The results, based on the (off-the shelf) default setting options for the monocular setting of ORB SLAM2 for the 07 and 10 sequences, are overall very accurate

with a low translation error. The following sections with different conditions will compare to the off-the-shelf results and explain how the performance will be impacted depending on each condition.

3.2 Number of Features Reduction

Based on the principles mentioned in section 2, the feature point extraction is closely related to the accuracy of camera pose estimation in the SLAM framework. The accuracy of the camera pose estimation can be improved when the features are evenly distributed and extracted from consecutive images. However, the position of the feature point extraction from the images is significantly sensitive and dependent on the image, which makes it difficult to extract by evenly distributing features. This section demonstrates how the estimated camera trajectory is impacted for three different levels of number of ORB features. This experiment used the KITTI dataset sequences to evaluate the estimated camera trajectory, so the camera, ORB, and viewer parameters can be controlled from KITTI04-12.yaml to evaluate the results. The evaluation method is only needed to change the number of ORB features, and the remaining parameters should be the same.

Table 2: The three different cases of the number of ORB features for evaluating the 07 KITTI sequence.

Case	(a)	(b)	(c)
Number of Features	1800	1500	1200

Table 2 shows the 3 different levels of the number of features applied and the naming convention is used in the paper. The cases are respectively evaluated for ATE using the 07 KITTI sequence as follows:

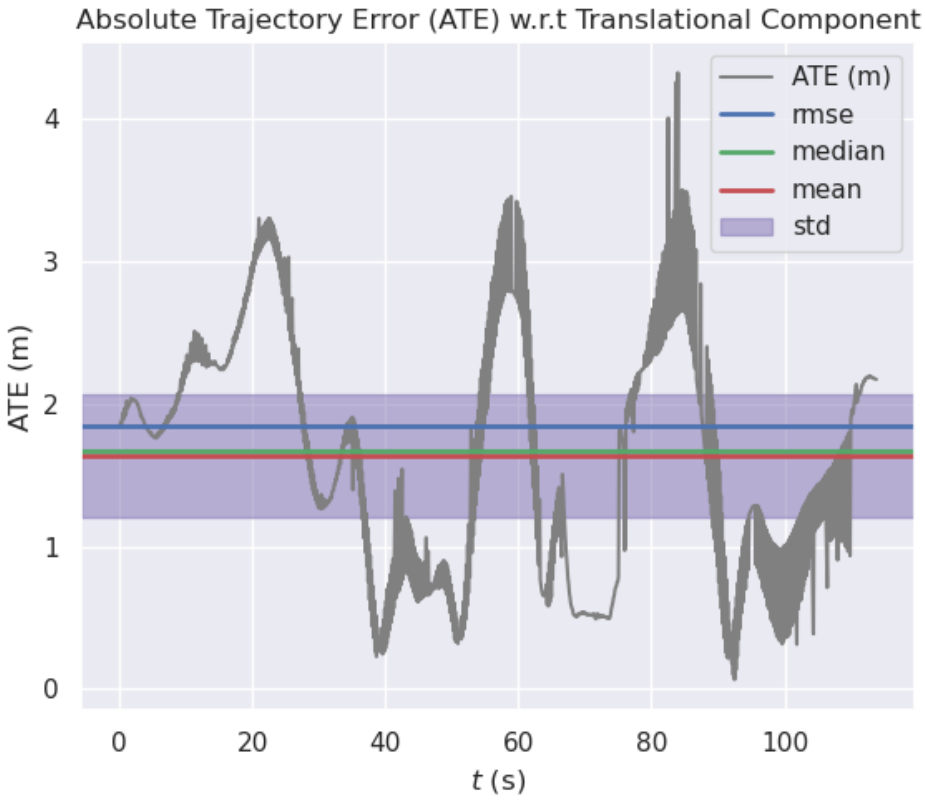

Figure 9: Evaluation of the absolute trajectory error between the ground truth trajectory and the estimated camera trajectory for **case (a)** using the 07 KITTI sequence.

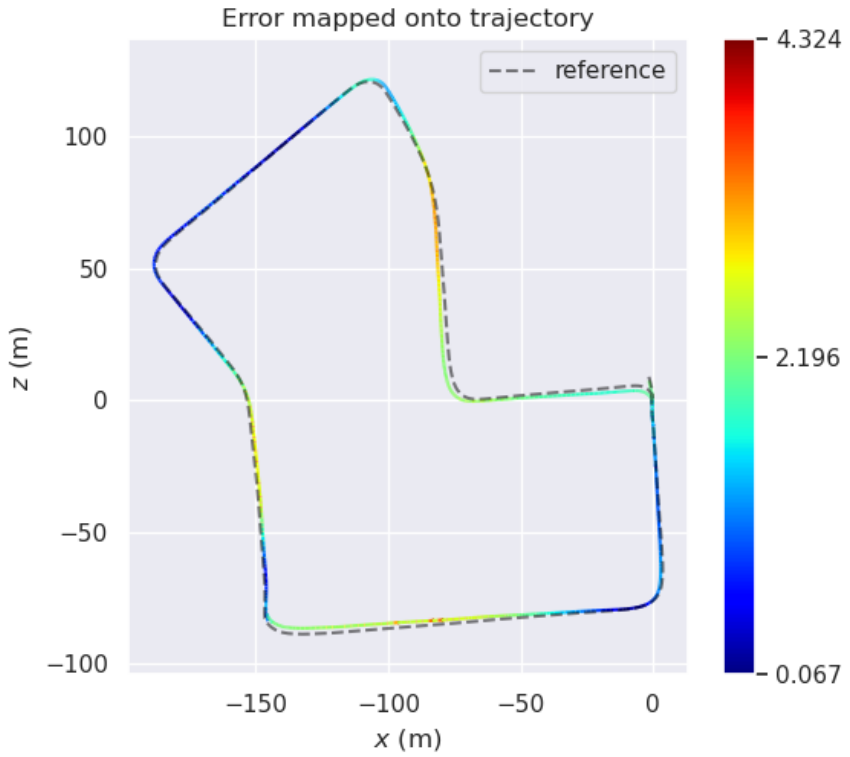

Figure 10: Evaluation of the error mapping between the ground truth trajectory and the estimated camera trajectory for **case (a)** using the 07 KITTI sequence.

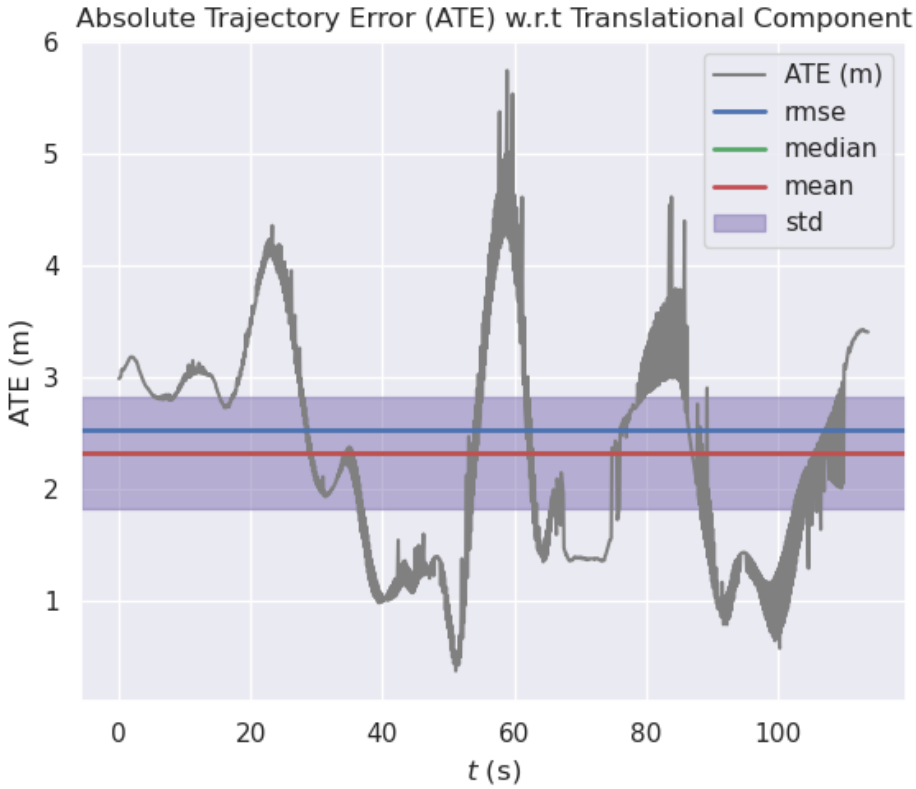

Figure 11: Evaluation of the absolute trajectory error between the ground truth trajectory and the estimated camera trajectory for **case (b)** using the 07 KITTI sequence.

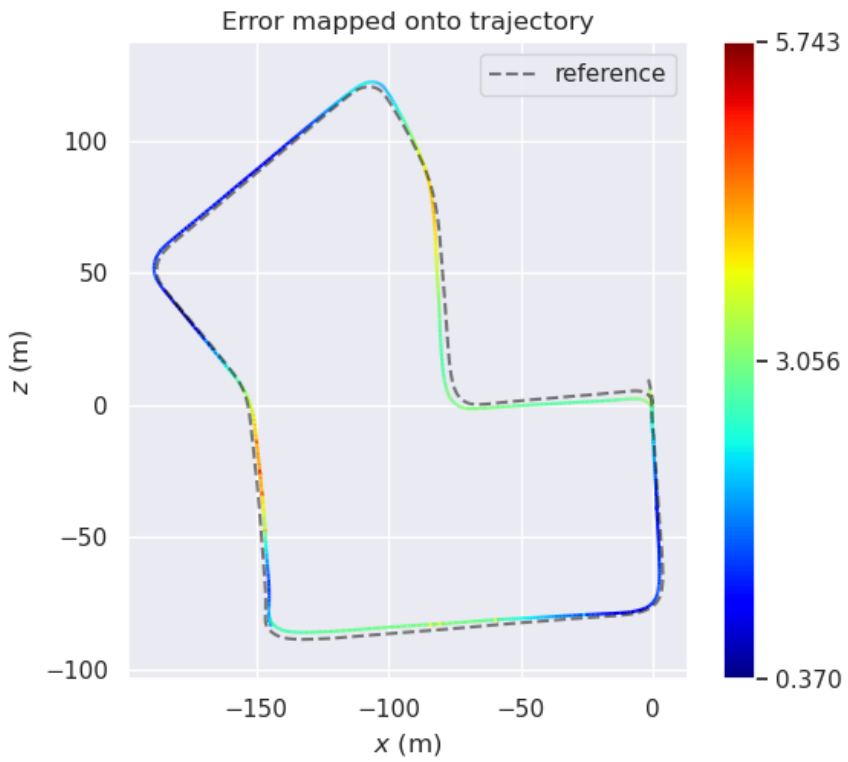

Figure 12: Evaluation of the error mapping between the ground truth trajectory and the estimated camera trajectory for **case (b)** using the 07 KITTI sequence.

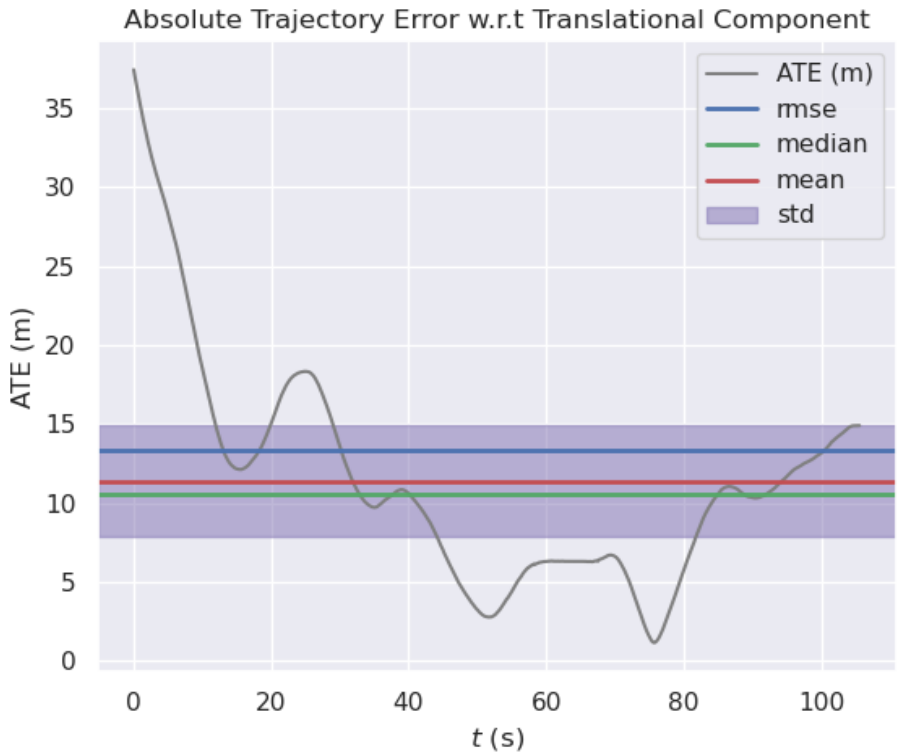

Figure 13: Evaluation of the absolute trajectory error between the ground truth trajectory and the estimated camera trajectory with **case (c)** using the 07 KITTI sequence.

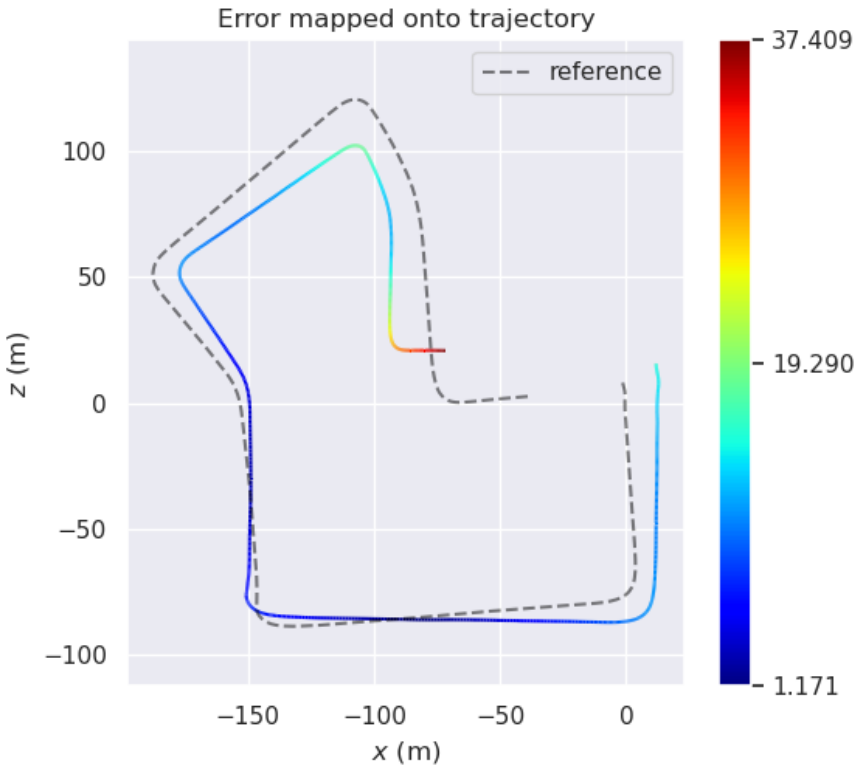

Figure 14: Evaluation of the error mapping between the ground truth trajectory and the estimated camera trajectory with **case (c)** using the 07 KITTI sequence.

As shown in **Figure 9**-**Figure 12**, the estimated camera trajectory with the 07 KITTI sequence for case (a) and (b) are considerably similar to the ground truth trajectory and has a small error. In addition, the colour bar next to the error maps indicate the value of ATE such that higher values are red lower values are blue. **Figure 9**-**Figure 12**, show that the locations in the path of highest and lowest ATE are considerably similar, but the ATE in case (b) is lower than in case (a) as expected. In other words, the lower number of features provided higher values of ATE. This means that it is difficult to convert them to 3D points due to the lower value of extracting the feature points, and then the error of the estimated camera trajectory increases. However, case (c) interestingly failed to map at an early stage as shown in (Agapito, Wang and Lu, 2021) . This means that the monocular system failed to extract the feature points from the images needed to determine the estimated camera trajectory under the condition in case (c). In the early stages, the system lost the path which failed to initialise the process as it was hard to compute which caused that the trajectory to significantly deviate from the ground truth trajectory during the sequence. Furthermore, case (c) failed to extract the feature points from the images at the early stages, so the system could not detect the same feature points, which means that it did not properly detect the loop at the end-stage.

Table 3: Evaluation of the metrics of the ATE on the 3 different cases of the number of ORB features with the 07 KITTI sequence.

Value (m)	07 KITTI sequence		
	Case (a)	Case (b)	Case (c)
RMSE of ATE	1.845766	2.532580	13.38271
SSE of ATE	3733.908	7023.290	182141.7
SD of ATE	0.856545	1.006192	7.023142
Maximum of ATE	4.324389	5.742696	37.40877
Minimum of ATE	0.067294	0.369663	1.171253
Mean of ATE	1.634987	2.324121	11.39178
Median of ATE	1.667935	2.317878	10.54904

Table 3 represents the value of each evaluation metrics of the ATE with 3 different cases on the 07 KITTI sequence. As mentioned in the previous section, if the metric values of ATE are bigger, then the error of the estimation camera trajectory increases, and the performance decreases. Case (a) seems to behave similarly to off-the-shelf option with low translation and rotation error. Case (b) is also accurate, but the ATE values are larger than case (a). Case (c) has huge values of ATE because it failed to map the sequence at an early stage, which means it is inaccurate with large translation and rotation error during the sequence. Therefore, if the number of ORB features is low, then the system cannot extract the feature points from the images properly and even failed to map the sequence. This problem results in failed loop detection that provides lower performance. If the number of features is less than 1000, the system failed to map, which is why the results are not presented.

Table 4: The three different cases of the number of ORB features for evaluating the 10 KITTI sequence.

Case	(a)	(b)	(c)
Number of Features	1800	1500	1200

Table 4 shows that 3 different levels of the number of features for the 10 KITTI sequence, as applied for the 07 KITTI sequence evaluation. The ATE and trajectory is evaluated for each case respectively.

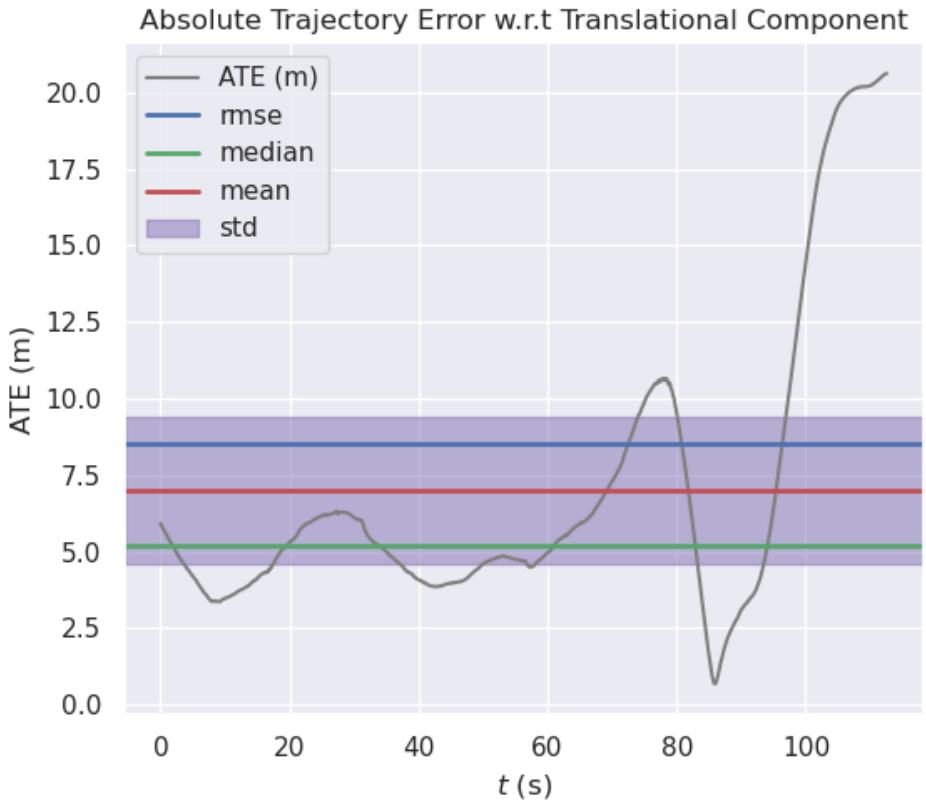

Figure 15: Evaluation of the absolute trajectory error between the ground truth trajectory and the estimated camera trajectory for **case (a)** using the 10 KITTI sequence.

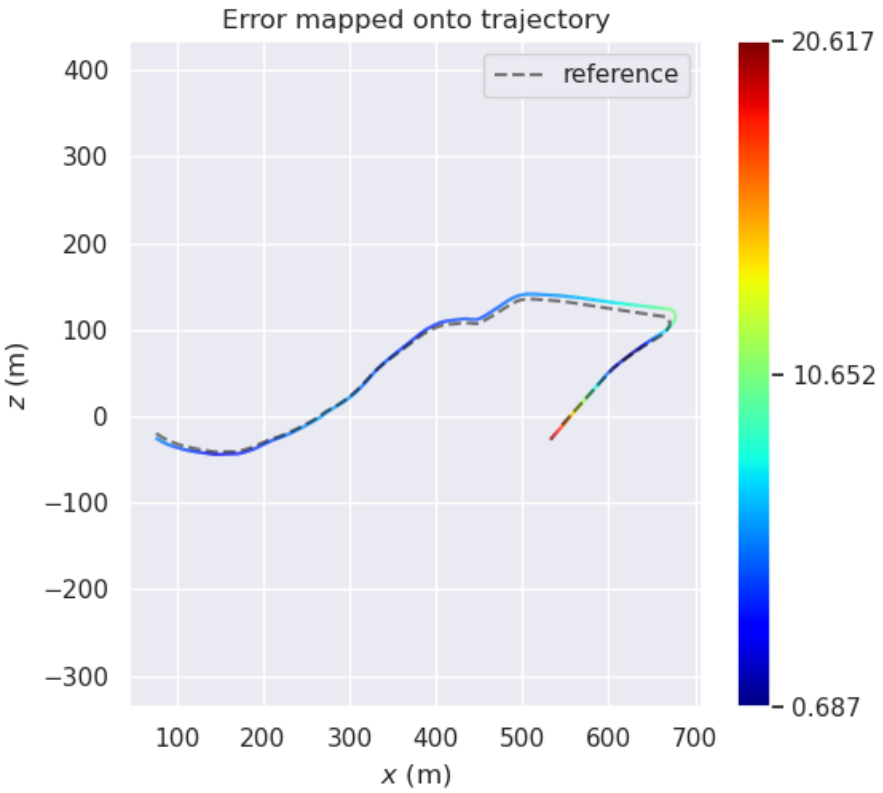

Figure 16: Evaluation of the error mapping between the ground truth trajectory and the estimated camera trajectory for **case (a)** using the 10 KITTI sequence.

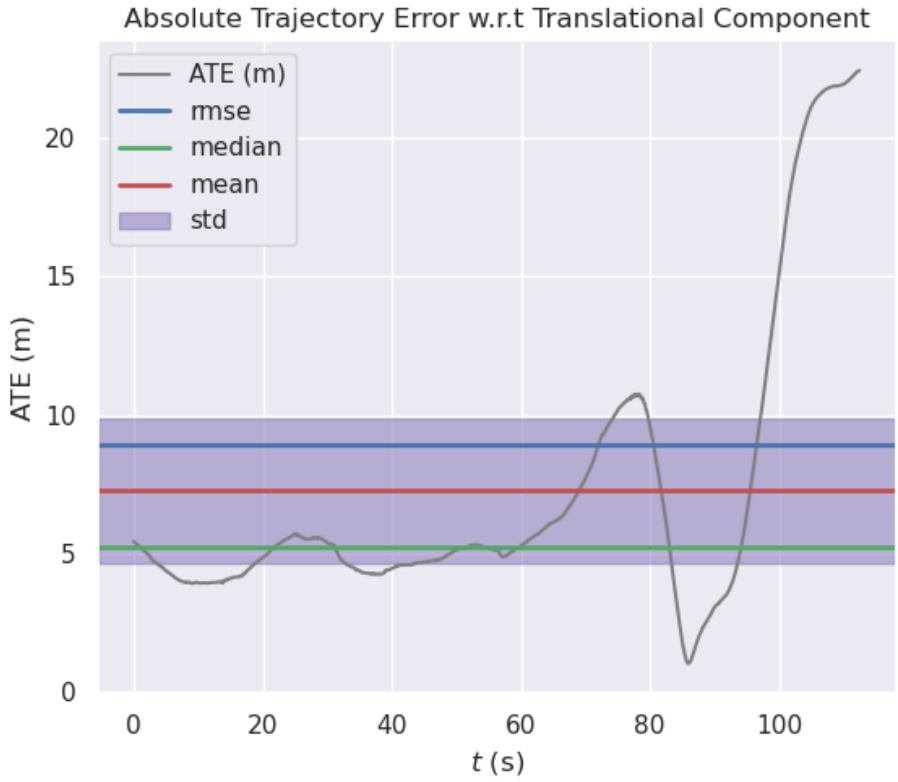

Figure 17: Evaluation of the absolute trajectory error between the ground truth trajectory and the estimated camera trajectory for **case (b)** using the 10 KITTI sequence.

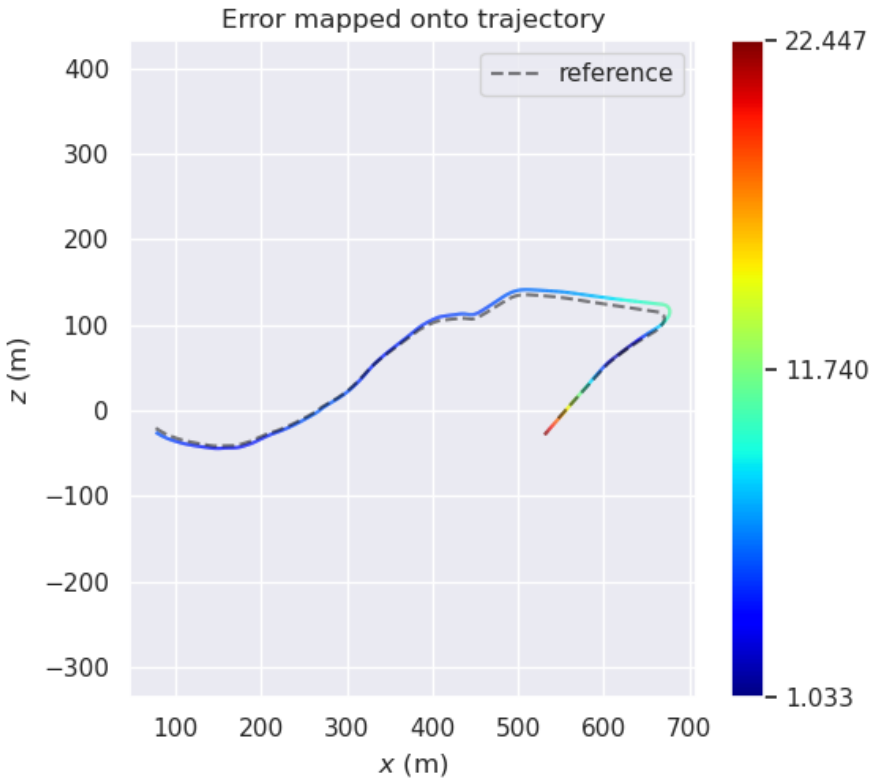

Figure 18: Evaluation of the error mapping between the ground truth trajectory and the estimated camera trajectory for **case (b)** using the 10 KITTI sequence

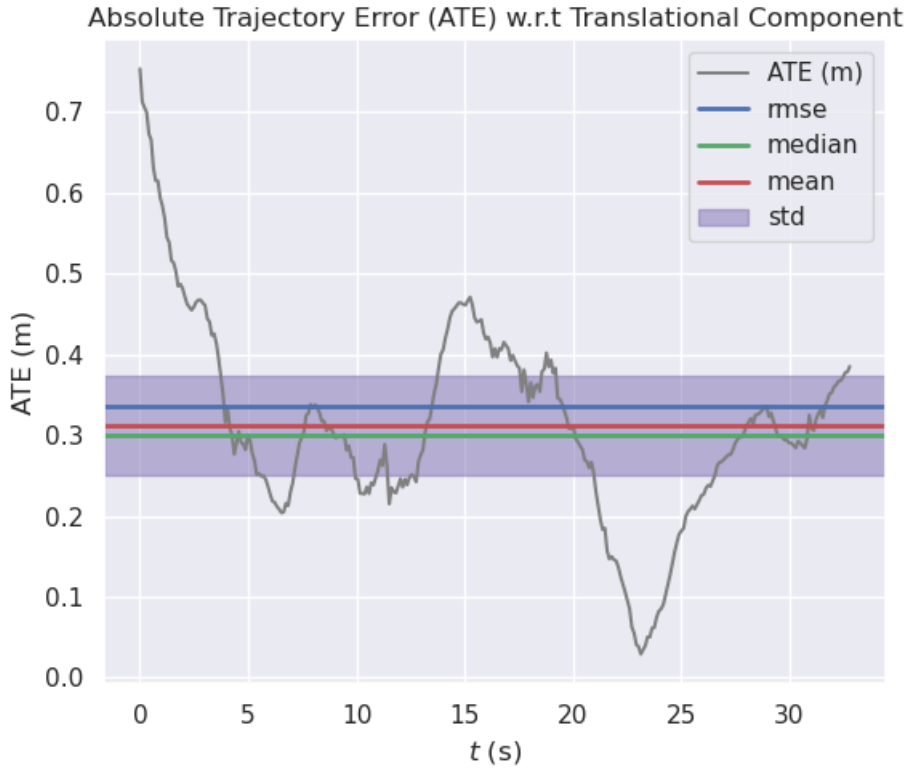

Figure 19: Evaluation of the absolute trajectory error between the ground truth trajectory and the estimated camera trajectory for **case (c)** using the 10 KITTI sequence

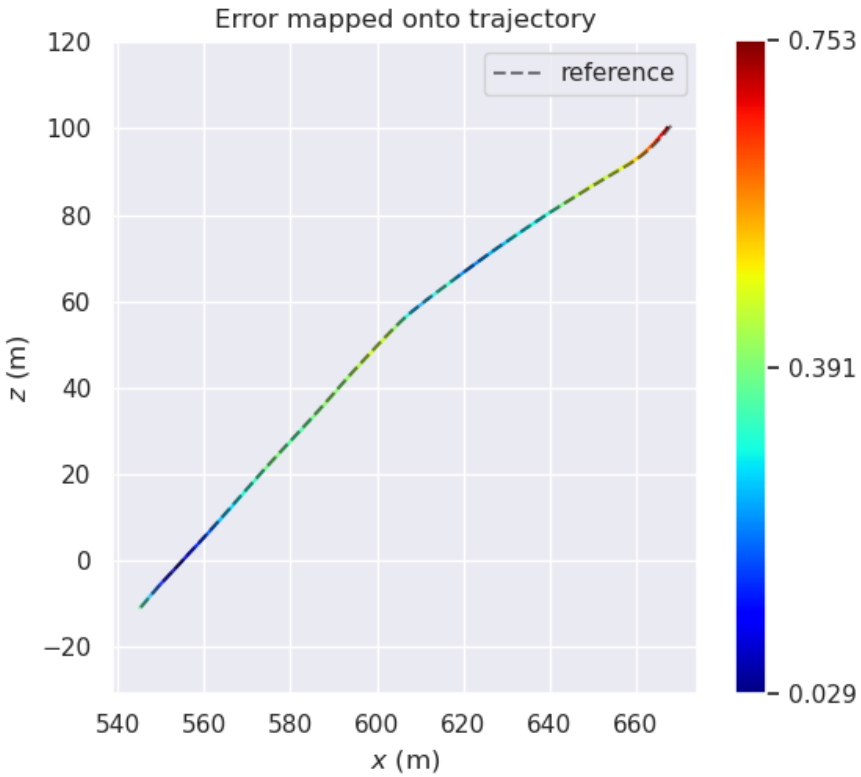

Figure 20: Evaluation of the error mapping between the ground truth trajectory and the estimated camera trajectory for **case (c)** using the 10 KITTI sequence.

The estimated camera trajectory along with the ground truth trajectory is represented in **Figure 16**, **Figure 18** and **Figure 20**, and it does not contain any loops. Similar to the results of the 07 KITTI sequence, the ATE increases for the 10 KITTI sequence when the number of features decreases, as expected. Furthermore, the value of ATE fluctuates significantly at drift paths, and the ATE increases as the distance increases with the large translation errors. **Figure 20** shows that the process totally failed to map during the sequence and could only extract the feature points after approximately 30 seconds into the sequence, which corresponded to approximately 540m of translation. As in the 07 KITTI sequence analysis, the system, using the 10 KITTI sequence. also failed to find the feature points when a low number of ORB features was applied and totally failed to map the sequence. This is because the monocular system has a single camera which makes it difficult to evenly distribute the feature points extracted. Furthermore, the path is winding which makes it difficult to compute the rotation and translation of the camera poses in real-time. It is hypothesized that if a lower number of features than case (c) is applied, the system will also fail to map.

Table 5: Evaluation of the metrics of the ATE on the 3 different cases of the number of ORB features with the 10 KITTI sequence.

Value (m)	10 KITTI sequence		
	Case (a)	Case (b)	Case (c)
RMSE of ATE	8.509523	8.948493	0.335536
SSE of ATE	78639.41	86881.95	35.68925
SD of ATE	4.818131	5.228281	0.122540
Maximum of ATE	20.61714	22.44673	0.753500
Minimum of ATE	0.687131	1.033296	0.029166
Mean of ATE	7.014100	7.262273	0.312359
Median of ATE	5.215915	5.210765	0.300579

Similarly to the results of the ATE metrics for the 07 KITTI sequence, the 10 KITTI sequence shows that that the larger metrics values of ATE are provided when the lower number of ORB features is applied, and the failure of the monocular system in case (c) can be seen numerically in **Table 5**, which is almost zero. Case (a) seems to be accurate with low translation and rotation error and is relatively similar to the values of ATE for the off-the-shelf option, which may suggest better performance. Case (b) is also accurate as similar metrics values of ATE are observed as in case (a), but slightly larger, so we can say that the performance in case (a) is better than case (b). On the other hand, the metrics of ATE in case (c) were provided with significantly small values because the system only extracted the feature points and initialised the process after the position reached approximately 540m, which corresponded to the straight path. As aforementioned, lowering the number of features would likely not provide any results.

For this section, three different cases with decreasing number of ORB features were applied to each KITTI sequence and the results evaluated. Selection of an appropriate number of features affects system performance significantly. Very low number of features in the monocular setting of the ORB SLAM system caused improper extraction of feature points and a lack of computation of trajectory (following the principles of section 2), which resulted in a failure of mapping the process. Furthermore, comparing the results on each sequence of the default setting (2000 number of features) in section 3.1 and section 3.2 (case (a), (b), (c)), the higher number of ORB features (default setting) provided the best performance in the system. Therefore, we need to input a higher number of ORB features in the monocular SLAM system to enable uniform distribution of feature points from the images and to get an accurate estimated camera trajectory.

3.3 Turning Off Outlier Rejection

Based on the principles in section 2, the optimisation method uses RANSAC to eliminate the outliers to obtain an accurate estimated camera trajectory. The feature points can be extracted through consecutive images and the motion of the camera can be expressed as through the fundamental matrix. However, the optimisation stage is used to fit the feature points such that they are aligned rather than the estimate the essential matrix, which has eight or nine parameters. For example, we can simply explain that the system automatically draws a line that estimates the parameters of a straight line with two points randomly among many feature points extracted from the image. An error function is computed by establishing a threshold corresponding to the straight line for how far the remaining points are and how far away from the estimated model. After that, the points out of the threshold are not considered anymore and only points inside of the threshold are considered for computation to optimise the process. This process can be optimised by iterating several times to extract the feature points randomly and use them to compute the estimated camera trajectory to obtain accurate information. The points contained in the threshold are called inliers, and the points that are not contained are outliers. If the system runs without removing outliers during the optimisation process and computes all the points (outliers + inliers), then we can easily expect that the estimated camera trajectory will obtain significantly skewed results from the ground truth trajectory and it might fail tracking (or at least suffer substantially), which means that the performance will be poor.

To execute this, some code within the structure of the ORB-SLAM2 system needs to be edited in order to turn off the outlier rejection stage. The edits are detailed below:

1. Line 397 in Optimizer.cc: pFrame→mvbOutlier[idx]=false;
2. Line 426 in Optimizer.cc: pFrame→mvbOutlier[idx]=false;

These two lines are in the Opimizer.cc file to perform the PoseOptimization. We turn the value from true to false so that the system cannot detect outliers. Thus, it does not remove the outliers and accurately classify inliers during the optimization. The results without the outlier rejection using each sequence can be represented as follows:

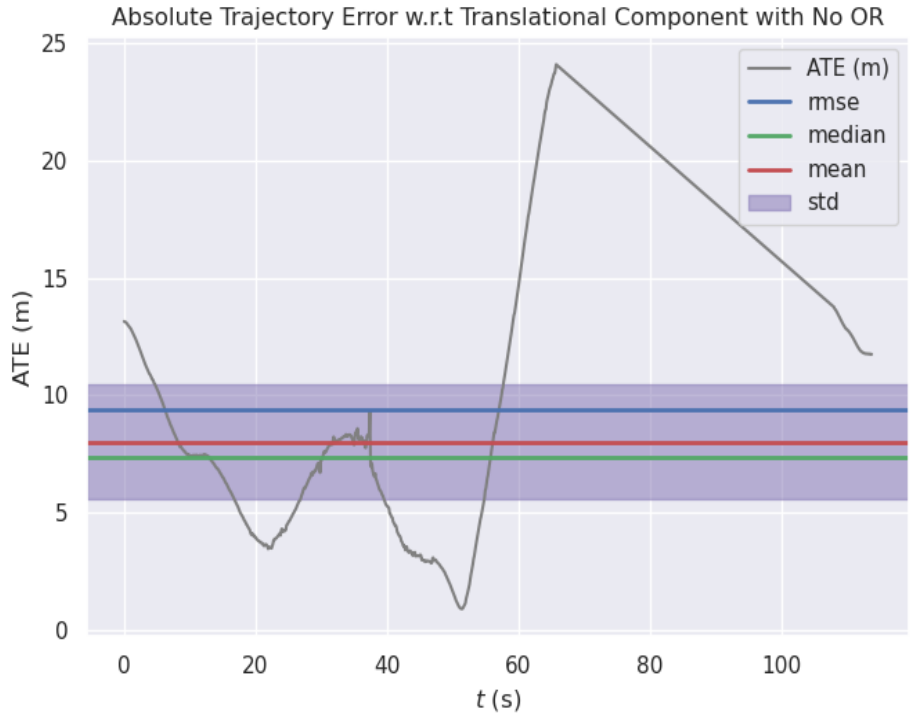

Figure 21: Evaluation of the absolute trajectory error between the ground truth trajectory and the estimated camera trajectory without outlier rejection using the 07 KITTI sequence.

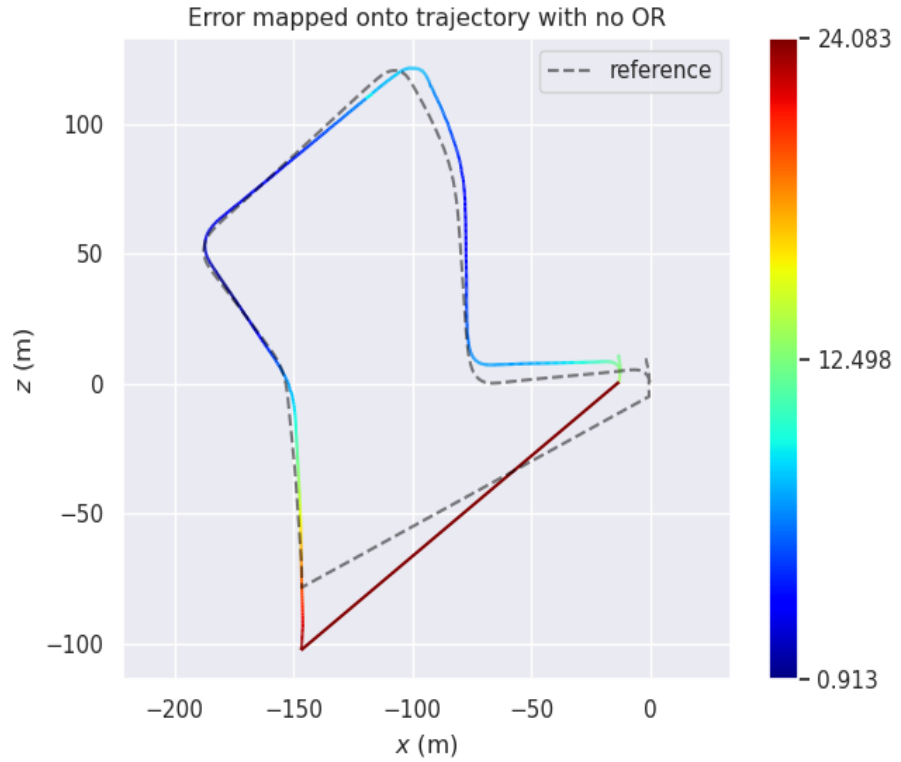

Figure 22: Evaluation of the error mapping between the ground truth trajectory and the estimated camera trajectory without outlier rejection using the 07 KITTI sequence.

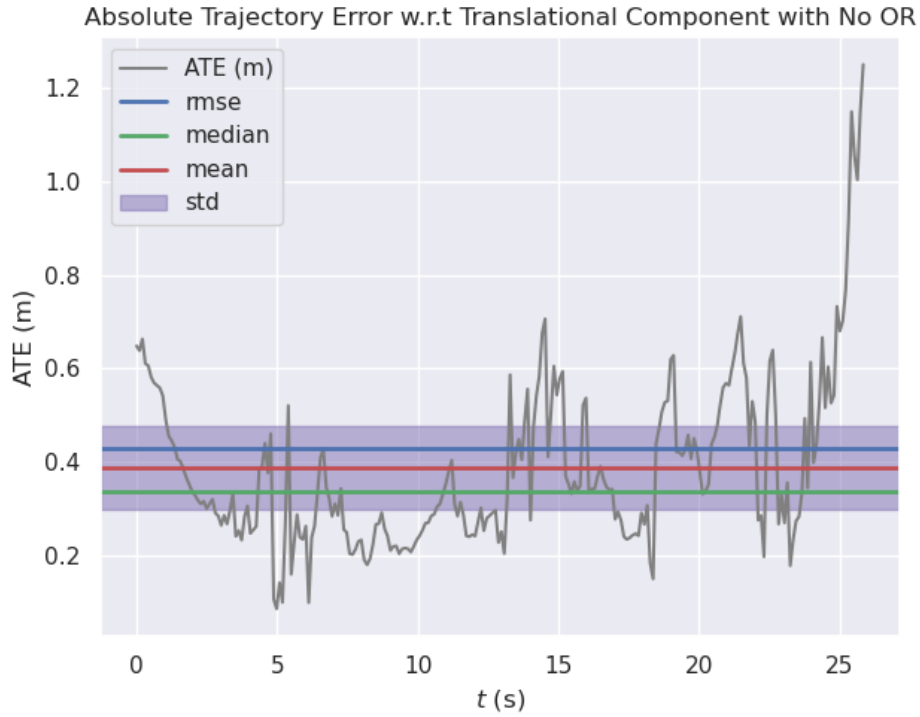

Figure 23: Evaluation of the absolute trajectory error between the ground truth trajectory and the estimated camera trajectory without outlier rejection using the 10 KITTI sequence.

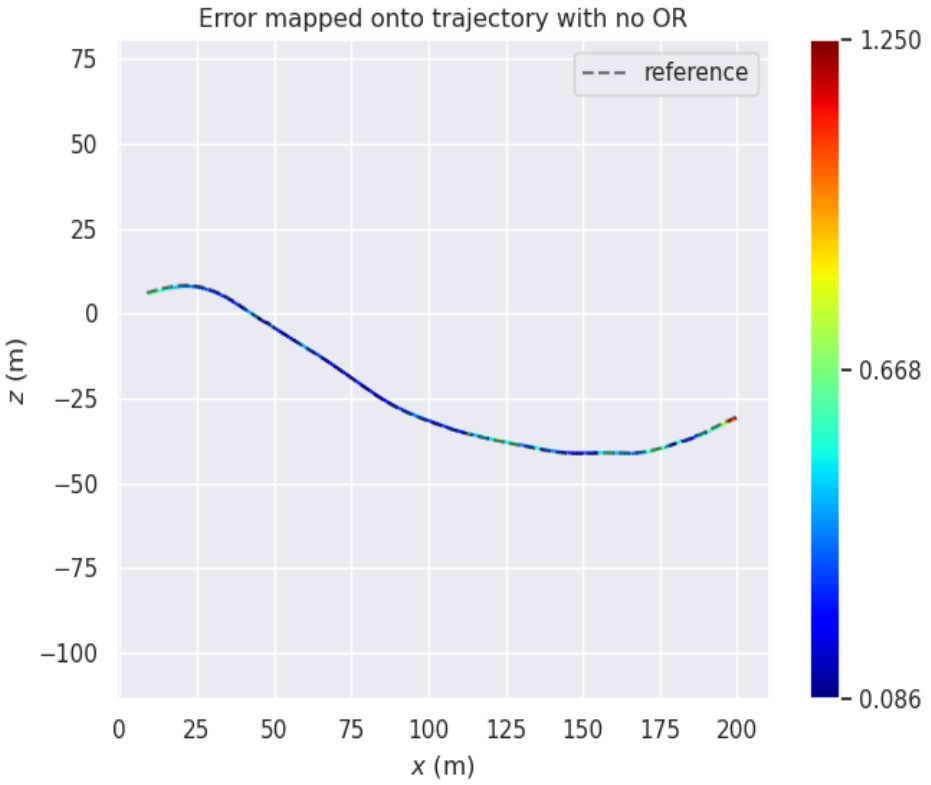

Figure 24: Evaluation of the error mapping between the ground truth trajectory and the estimated camera trajectory without outlier rejection using the 10 KITTI sequence.

Table 6: Evaluation of the metrics of the ATE on the 07 and 10 KITTI sequence without outlier rejection.

Value (m)	KITTI sequence	
	07	10
RMSE of ATE	9.370500	0.427723
SSE of ATE	60586.33	45.73668
SD of ATE	4.839008	0.180468
Maximum of ATE	24.08317	1.249865
Minimum of ATE	0.912558	0.085747
Mean of ATE	8.024355	0.387786
Median of ATE	7.387589	0.337248

Once outlier rejection was disabled, for both 07 and 10 KITTI sequences the trajectories are significantly affected as shown in **Figure 22** and **Figure 24**. The reason behind that is that the outlier features contribute to the error during the estimation due to its abnormal value, as expected.

Compared to the 07 sequence KITTI with default setting in **Figure 5** and **Figure 6**, incorporating the outliers leads to a much higher ATE error. When evaluating the error mapping as shown in **Figure 22**, there is a deviation from the ground truth trajectory particularly in the area on the plot between (-150, -100) to (0, 0). This means that the system accumulated significant error form the starting point (0, 0) to the end point (-150, -100) until the SLAM system accounted for the error by detecting previously seen errors and performing BA. In other words, It is hypothesized that the red line beginning at (-150, -100) would be far more incorrect without the loop closure optimiser enabled, which aims to detect and build a loop. Therefore, the loop closure helps to build this red straight line to close the loop. Numerically, this caused an increase in the mean ATE of ~450% as shown in **Table 6**, which is evidently quite significant.

At first sight, when comparing the 10 KITTI sequence with default settings (as in **Figure 7** and **Figure 8**) to the outlier rejection being turned off, the ATE and the trajectory does not seem to be adversely affected. This is because the tracking is lost at point (200, -25) and t = 25s which causes **Figure 23** and **Figure 24** to seem reasonable. In reality, the ground truth trajectory ends at point (550, 0) and t = 200s. **Table 6**'s results are also misleading due to the same notion. Since the 10 KITTI sequence does not have a loop, loop closure is not working here to improve the estimated trajectory. This caused the plots creation to terminate at the point where the system lost the estimated camera poses. In other words, this was a full failure of the SLAM system.

3.4 Turning Off Loop Closure

The principle of loop closure is that if the system comes back to the same location, it can detect the same feature points during a sequence, then the system automatically connects the trajectory between the matching points. Therefore, it is hypothesized that the system cannot properly connect the trajectory when it comes back to the same location when the loop closure is switched off due to error accumulation in camera pose estimation. In other words, the system cannot properly build a loop during the sequence.

Once again, to turn off loop closure, some code within the structure of the ORB-SLAM2 system needs to be edited. The edits are detailed below:

1. Line 1522 from Tracking.cc: mpLoopClosing→RequestReset();
2. Line 87 from LocalMapping.cc: mpLoopCloser→InsertKeyFrame(mpCurrentKeyFrame);
3. Line 94 from System.cc: mpLoopCloser = new LoopClosing(mpMap, mpKeyFrameDatabase, mpVocabulary, mSensor != MONOCULAR);
4. Line 95 from System.cc: mptLoopClosing = new thread(&ORB_SLAM2::LoopClosing::Run, mpLoopCloser);
5. Line 107 from System.cc: mpTracker→SetLoopClosing(mpLoopCloser);
6. Line 110 form System.cc: mpLocalMapper→SetLoopCloser(mpLoopCloser);

7. Line 112 from System.cc: mpLoopCloser→SetTracker(mpTracker);
8. Line 113 from System.cc: mpLoopCloser→SetLocalMapper(mpLocalMapper);
9. Line 304 from System.cc: mpLoopCloser→RequestFinish();
10. Line 313 from System.cc: while(!mpLocalMapper→isFinished())

Step 1-9 were all commented out to switch off the loop closure, and only in step 10 were there changes because !mpLoopCloser→isFinished(), and mpLoopCloser→isRunningGBA() consist of the part of the loop closure, so these codes are not considered anymore and only local mapping part should be used in this section. Finally, the figures which evaluate the ATE and trajectory error mapping during each sequence is shown below.

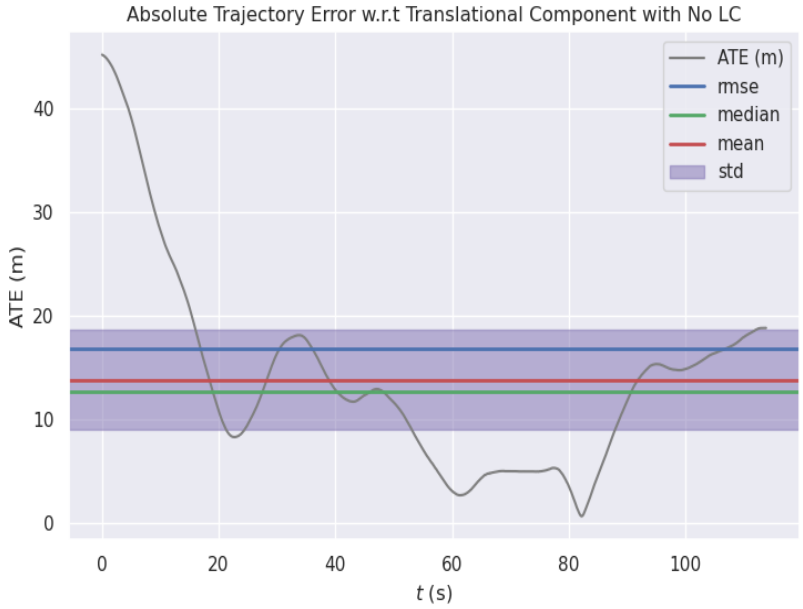

Figure 25: Evaluation of the absolute trajectory error between the ground truth trajectory and the estimated camera trajectory without loop closure using the 07 KITTI sequence.

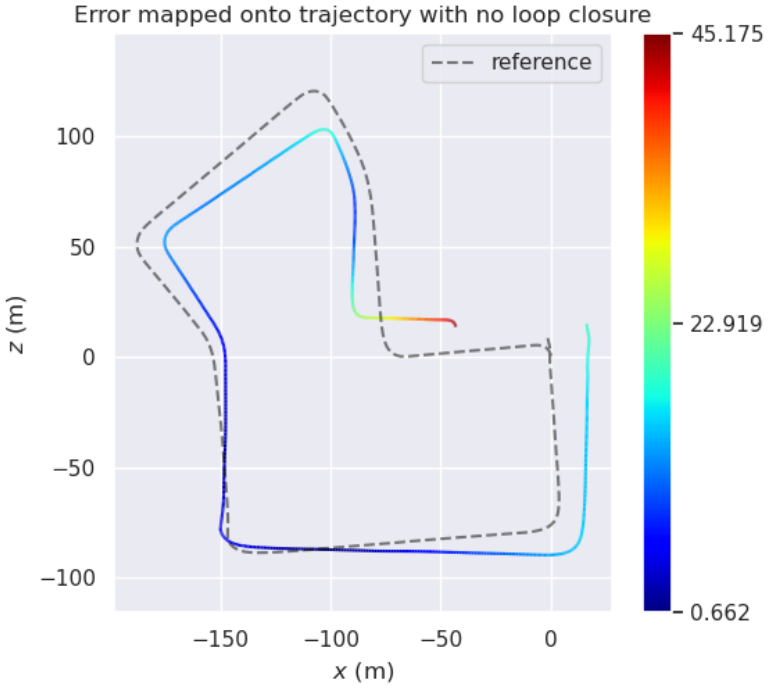

Figure 26: Evaluation of the error mapping between the ground truth trajectory and the estimated camera trajectory without loop closure using the 07 KITTI sequence.

Figure 27: Evaluation of the absolute trajectory error between the ground truth trajectory and the estimated camera trajectory without loop closure using the 10 KITTI sequence.

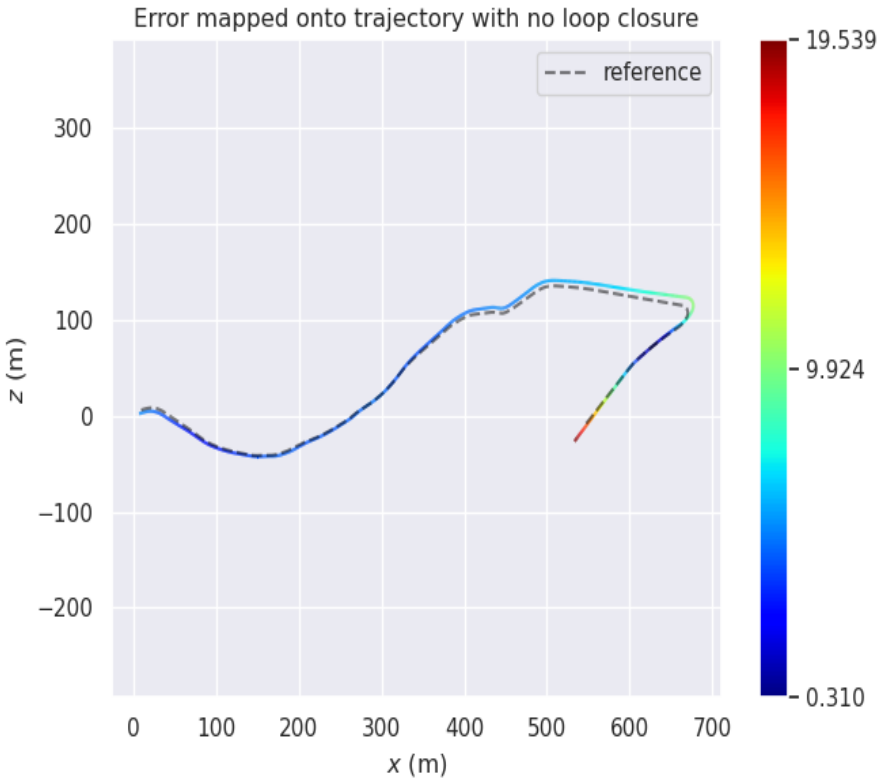

Figure 28: Evaluation of the error mapping between the ground truth trajectory and the estimated camera trajectory without loop closure using the 10 KITTI sequence.

Table 7: Evaluation of the metrics of the ATE on the 07 and 10 KITTI sequence without loop closure.

Value (m)	KITTI sequence	
	07	10
RMSE of ATE	16.82663	7.882488
SSE of ATE	310316.5	72696.34
SD of ATE	9.570812	4.422015
Maximum of ATE	45.17546	19.53885
Minimum of ATE	0.661790	0.309838
Mean of ATE	13.83962	6.525290
Median of ATE	12.70341	4.807168

Disabling loop closure caused the 07 KITTI sequence to be significantly affected as shown in **Figure 25** and **Figure 26** compared to the original plots. However, 10 KITTI sequence seems to be unaffected as shown in **Figure 27** and **Figure 28** compared to the original plots.

The reason why the performance of 07 KITTI sequence is worse than before is that the main job of the loop closure is to form a closed loop and correct the poses with optimization to keep the estimation on track. When we turned off the loop closure, we can clearly see that in **Figure 26**, the map of the trajectory is not closed. The gap between point (0, 0) and (50, 0) is caused by the system initialization which would waste some time.

The reason why the performance of 10 KITTI sequence is unaffected is that 10 KITTI sequence does not contain a loop and therefore does not perform loop closure during the estimation. Numerically, as shown in **Table 7**, the mean ATE increased by 847% and 1% for the 07 and 10 KITTI sequences respectively, which are expected results due to the aforementioned reasons.

4 Conclusion

This paper initially presented the tracking thread of the ORB-SLAM2 system in detail. ORB-SLAM2 is one of the most robust and commonly used visual SLAM system. The steps for predicting the camera pose and performing localisation was shown for a monocular assumed system where the camera intrinsics, such as the focal length, are known. The next section of the paper showed how the ORB-SLAM2 system performed when analysing the system with off-the-shelf settings, reducing the number of ORB features, disabling outlier rejection, and disabling loop closure.

To evaluate the SLAM system, absolute trajectory error (ATE) is used as the relevant metric to compare the estimated and ground truth trajectories. In this report, two sequences were tested, the 07 KITTI sequence and the 10 KITTI sequence, the former includes a loop in the trajectory and the latter is a winding path with significant rotation. For the off-the-shelf settings, with 2000 ORB features and all optimisers activated, the system performed very well achieving low ATE. Reduction of the number of ORB features caused a steady increase in ATE up until 1200 features, when the system was unable to map or localise. Disabling outlier rejection caused the largest effect on the performance of the system for the 10 KITTI sequence and had the highest ATE. For the 07 KITTI sequence, the disabling of outlier rejection still affected the performance but did not have as much of an impact as disabling loop closure. Disabling loop closure vastly affected the performance in the 07 KITTI sequence, but did not have a significant effect on the 10 KITTI sequence as it does not have a loop in the trajectory.

References

Agapito, L. (2021) 'Robotic Vision and Navigation Part III Visual SLAM Lecture Slides', *COMP0130 Lecture Notes*.

Agapito, L., Wang, J. and Lu, Z. (2021) 'Coursework Description', *COMP0130 Module*.

Geiger, A., Lenz, P. and Urtasun, R. (2012) 'Are we ready for autonomous driving? the KITTI vision benchmark suite', *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.

Horn, B. K. P., Hilden, H. M. and Negahdaripour, S. (1988) 'Closed-form solution of absolute orientation using orthonormal matrices', *Journal of the Optical Society of America A*, 5(7), p. 1127. doi: 10.1364/josaa.5.001127.

Klein, G. and Murray, D. (2007) 'Parallel tracking and mapping for small AR workspaces', *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR*, pp. 225–234. doi: 10.1109/ISMAR.2007.4538852.

Mur-Artal, R., Montiel, J. M. M. and Tardos, J. D. (2015) 'ORB-SLAM: A Versatile and Accurate Monocular SLAM System', *IEEE Transactions on Robotics*, 31(5), pp. 1147–1163. doi: 10.1109/TRO.2015.2463671.

Mur-Artal, R. and Tardos, J. D. (2017a) 'ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras', *IEEE Transactions on Robotics*, 33(5), pp. 1255–1262. doi: 10.1109/TRO.2017.2705103.

Mur-Artal, R. and Tardos, J. D. (2017b) 'Visual-Inertial Monocular SLAM with Map Reuse', *IEEE Robotics and Automation Letters*, 2(2), pp. 796–803. doi: 10.1109/LRA.2017.2653359.

Rosten, E. and Drummond, T. (2006) 'Machine learning for high-speed corner detection', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS(July 2006), pp. 430–443. doi: 10.1007/11744023_34.

Rublee, E. *et al.* (2011) 'ORB: An efficient alternative to SIFT or SURF', *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2564–2571. doi: 10.1109/ICCV.2011.6126544.

Silveira, G., Malis, E. and Rives, P. (2008) 'An efficient direct approach to visual SLAM', *IEEE Transactions on Robotics*, 24(5), pp. 969–979. doi: 10.1109/TRO.2008.2004829.

Sturm, J. *et al.* (2012) 'A Benchmark for the Evaluation of RGB-D SLAM Systems', *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 573–580. doi: 10.1109/IROS.2012.6385773.

Younes, G. *et al.* (2017) 'Keyframe-based monocular SLAM: design, survey, and future directions', *Robotics and Autonomous Systems*, 98, pp. 67–88. doi: 10.1016/j.robot.2017.09.010.