TQU-SLAM Benchmark Feature-based Dataset for Building Monocular VO

Van-Hung Le
Tan Trao University, Tuyen Quang, 22000, Vietnam
van-hung.le@mica.edu.vn (corresponding author)

Huu-Son Do
Tan Trao University, Tuyen Quang, 22000, Vietnam
dosonhytq@gmail.com

Van-Nam Phan
Tan Trao University, Tuyen Quang, 22000, Vietnam
Nampv0903@gmail.com

Trung-Hieu Te
Tan Trao University, Tuyen Quang, 22000, Vietnam
ttrunghieu97@gmail.com

Received: 24 April 2024 | Revised: 12 May 2024 | Accepted: 26 May 2024

 DOI: https://doi.org/10.48084/etasr.7611

ABSTRACT

This paper introduces the TQU-SLAM benchmark dataset, which includes 160,631 RGB-D frame pairs with the goal to be used in Dell Learning (DL) training of Visual SLAM and Visual Odometry (VO) construction models. It was collected from the corridors of three interconnected buildings with a length of about 230 m. The ground-truth data of the TQU-SLAM benchmark dataset, including the 6-DOF camera pose, 3D point cloud data, intrinsic parameters, and the transformation matrix between the camera coordinate system and the real world, were prepared manually. The TQU-SLAM benchmark dataset was tested based on the PySLAM framework with traditional features, such as SHI_TOMASI, SIFT, SURF, ORB, ORB2, AKAZE, KAZE, and BRISK and features extracted from DL LIKE VGG. Experiments were also conducted on DPVO for VO estimation. The camera pose estimation results were evaluated and presented in detail, while the challenges of the TQU-SLAM benchmark dataset were analyzed.

Keywords-TQU-SLAM benchmark dataset; Visual Odometry; RGB-D images; 3D trajectory; feature-based extraction

I. INTRODUCTION

Visual Odometry (VO) is applied in many fields, namely Autonomous Vehicles, Unmanned Aerial Vehicles (UAV), Underwater Robots, Space Explore Robots, Agriculture Robots, Medical Robots, and AR/VR [1]. The problem of estimating VO using a vision-based approach with monocular camera data is solved through the deployment of two approaches: geometry-based methods/knowledge-based and learning-based methods [2-4]. Learning-based methods are based on traditional Machine Learning (ML) and DL. In particular, DL has been applied to solve many ML problems, such as modeling groundwater storage change [4], predicting climate change [5], air pollution prediction [6], classification of nanosatellite images [7], automatic weed detection system using UAV images [8], forest area classification using UAV images [9], etc. The knowledge-based methods include appearance-based methods, feature-based methods, and hybrid methods. The framework to build a VO system entails five steps, with input data images: feature detection, feature tracking, motion estimation, triangulation, and trajectory estimation [2], as shown in Figure 1. The output of the VO framework is 6-DOF, involving the 3D pose, which is the position of the camera in the scene, while the remaining 3D is the direction of camera's movement.

Established on the feature-based methods, the feature detection step is the process of feature extraction (SIFT [10], SURF [11], ORB [12], BRISK [13] features, etc.). To find corresponding points on frames employing techniques such as optical flow [14]. This is followed by a motion estimation step that typically utilizes Epipolar geometry constraints to incorporate feature-to-feature matching (2D to 2D). From there, motion parameters are calculated and projection from 3D

to 2D is performed to minimize the 3D tracked landmarks against the current image frame and 3D to 3D to increase the accuracy of the estimated pose. To perform camera pose optimization, research often applies algorithms, such as Bundle adjustment, Kalman Filter, and EKF Graph optimization [1]. For DL-based approaches, the steps of feature detection, feature matching, and pose estimation are performed with DL algorithms [1].

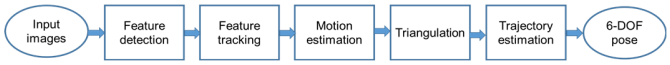

Fig. 1. Building a VO framework from input images.

Currently, most VO estimation methods are evaluated on the KITTI [15-17], TUM RGB-D SLAM [18], NYUDepth [19], and ICL-NUIM [20] datasets. KITTI 2012 [16] dataset is collected from two high-resolution camera systems, a Velodyne HDL64E laser scanner (grayscale and color), and a state-of-the-art OXTS RT 3003 localization system (a combination of devices, such as GPS, GLONASS, security IMU, and RTK correction signals). This dataset is divided into sub sets serving different problems. The dataset deployed to evaluate the optical flow estimation model includes 194 image pairs for training and 195 image pairs for testing, with a resolution of 1240×376 pixels. The dataset utilized to evaluate the 3D VO/SLAM model consists of 22 stereo sequences collected from a length of 39.2 km of driving. This data set provides benchmarks and evaluation measures for VO and Visual SLAM, such as motion trajectory and driving speed. In the dataset used to evaluate object detection and 3D orientation estimation, the original data include accurate 3D bounding boxes for object classes, such as cars, vans, trucks, pedestrians, cyclists, and trams. Original data of 3D objects in point cloud data were manually labeled to evaluate algorithms for 3D orientation estimation and 3D tracking. TUM RGB-D SLAM dataset was collected employing the MS Xbox Kinect sensor [18]. The collected data consist of RGB-D frame sequences. The environment for data collection includes two different indoor scenes, the first is a typical office environment called "fr1" with a size of 6×6 m^2, and the second is a large industrial hall called "fr2" with a size of 10×12 m^2. The ground truth trajectory from the motion capture system is provided by eight high-speed tracking cameras. This dataset entails 39 frame sequences and is divided into four groups. Color and depth image data are 640×480 pixels in size and captured at a rate of 30 Hz. NYUDepth dataset [19] consists of 1449 RGB-D images collected from MS Kinect from multiple buildings in three US cities with 464 different indoor scenes belonging to 26 scene classes. The dataset contains 35,064 distinct objects, spread across 894 different classes. For each of the 1449 images, supporting captions were added manually. ICL-NUIM dataset [20] is a dataset consisting of RGB-D sequences used to evaluate VO, 3D reconstruction, and SLAM algorithms that were collected from a living room and an office room. For each scene the authors collected four frame sequences with different number of frames in each sequence. Original camera trajectories (POVRay) and synthetic trajectory data were obtained from ground truth depth maps and color images. In this dataset, there are two main types of noise: noise from color images and noise from depth images when collecting data with MS Kinect.

However, DL methods always need a very large amount of data to train a VO estimation model. With this approach, the more the data and data contexts are, the more accurate the VO estimation results are. In this paper, a standard dataset named the TQU-SLAM benchmark dataset is proposed to evaluate VO estimation methods. The employed database is collected with the use of the Intel RealSense D435 in an environment that is the 2nd floor of three interconnected buildings. The dataset includes 160,631 RGB-D frame pairs.

To see the challenges of the TQU-SLAM benchmark dataset, the proposed dataset was tested by the PySLAM [21] with the features extracted in the feature extraction step as ORB [22], ORB2 [23], SIFT [10], SURF [11], BRISK [12], AKAZE [24], KAZE[24], or features extracted from VGG DL model [25]. The current paper has the following main contributions:

- The TQU-SLAM benchmark dataset is introduced for evaluating VO models, algorithms, and methods. The ground truth of the TQU-SLAM benchmark dataset is constructed by hand including the camera coordinates in the real-world coordinate system, 3D point cloud data, intrinsic parameters, and the transformation matrix between the camera coordinate system and the real world.
- Experiments were carried out on the TQU-SLAM benchmark dataset for estimating VO based on PySLAM [21, 26] with extracted features, such as ORB [22], ORB2 [23], SIFT [10], SURF [11], BRISK [13], AKAZE [24], KAZE[24] , or features extracted from a DL model such as VGG [25].
- The results are analyzed and compared with the ones of the TQU- SLAM benchmark dataset.

II. TQU-SLAM BENCHMARK DATASET

A. Data Collection

The experiment is set up in the second-floor hallway of Building A, Building B, and Building C of Tan Trao University (TQU), Vietnam (Figure 2-3). The data were collected from the environment using an Intel RealSense D435 camera. The camera was mounted on a vehicle (Figure 4). The angle between the camera's view and the ground is about 45°. The total distance traveled by the vehicle at one time is FO-D = 230.63 m, OP-D = 228.13 m and the width is 2 m. For every 0.5 m, a numbered marker with dimensions of 10 × 10 cm was assigned for one marked corner. The total number of markers used was 332.

The moving speed when collecting data was about 0.2 m/s. The data collected are color and depth images with a resolution of 640 × 480 pixels. The car was always driven in the middle of the hallway. The data acquisition speed is 15 fps. Data collection was performed four times. On the first day, data were collected twice (1st, 2nd), and on the second day, data were collected the remaining two times (3rd, 4th). Data were collected in the afternoon at 2:00 p.m. and 3:00 p.m. Each time, the direction of movement according to the blue arrow is in the

forward direction (FO-D), and the direction of movement according to the red arrow is in the opposite direction (OP-D). All data of the TQU-SLAM benchmark dataset can be seen in [26]. Table I provides a summary of the collected data.

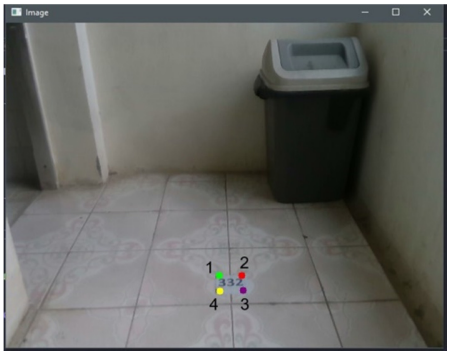

Fig. 2. Marker application and marker results collected on a color image.

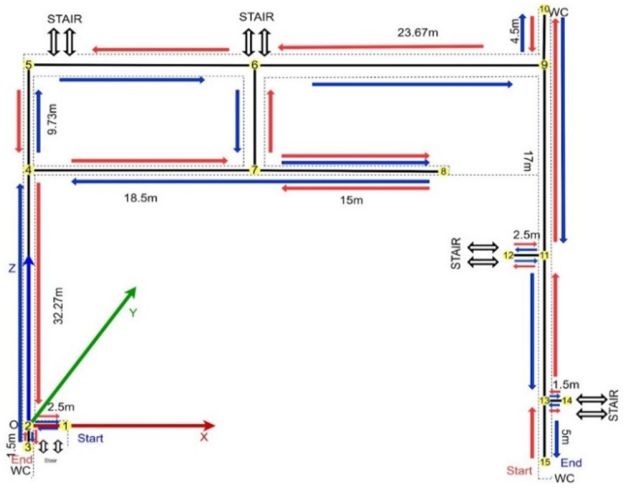

Fig. 3. Illustration of the hallway environment of Building A, Building B, and Building C of Tan Trao University, Vietnam for data collection. In the environment, 15 key points are highlighted. The direction of movement according to the blue arrow is in the forward direction, and the direction of movement according to the red arrow is in the opposite direction.

Fig. 4. Illustration of a vehicle equipped with a camera and computer when collecting data.

B. Preparing Ground-Truth Trajectory for Visual Odometry

To prepare the ground-truth data for evaluating the results of VO, the ground-truth data of the motion trajectory was built as follows. A coordinate system was pre-defined in real-world space as portrayed in Figure 3, where the X axis is red, the Y axis is green, and the Z axis is blue.

A self-developed tool was used in the Python programming language to mark four points on the color image, as depicted in Figure 5. Then the corresponding four marker points on the depth image were taken because each frame was obtained as a pair of RGB images and depth images. To convert the four marked points of the marker on the RGB image and the four corresponding points on the depth image to 3D point cloud data, the camera's intrinsic parameters shown in (1) were employed:

$$\begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 525.0 & 0 & 319.5 \\ 0 & 525.0 & 239.5 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where fx, fy, cx, and cy are the intrinsic parameters of the camera. For each marker point with coordinates *xd*, *yd*, and depth value da on the depth image, the result will be converted to point M with coordinates *xm*, *ym*, and *zm*:

$$xm = \frac{(\mathrm{xd} - \mathrm{cx}) \times \mathrm{da}}{\mathrm{fx}}$$
$$ym = \frac{(\mathrm{yd} - \mathrm{cy}) \times \mathrm{da}}{\mathrm{fy}} \quad (2)$$
$$ym = \mathrm{da}$$

TABLE I. NUMBER OF FRAMES

Data acquisition time	Direction	Number of RGB-D frames
1st	FO-D	21,333
	OP-D	22,984
2nd	FO-D	19,992
	OP-D	21,116
3rd	FO-D	17,995
	OP-D	20,814
4th	FO-D	17,885
	OP-D	18,548

As observed in Figure 5, the four points of the marker point cloud data are according to the camera coordinate system, with the original coordinate system being the center of the camera. Therefore, to find these four points in the real-world coordinate system, it is necessary to find the rotation and translation matrix (transformation matrix), to transform the four points from the camera coordinate system to the real-world coordinate system. This study assumes calling point M (*x*, *y*, *z*) in the camera coordinate system, and (*x*', *y*', *z*') in the real world coordinate system after performing the transformation according to (3):

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} Ro_{11} & Ro_{12} & Ro_{13} & Tr_1 \\ Ro_{21} & Ro_{22} & Ro_{23} & Tr_2 \\ Ro_{31} & Ro_{32} & Ro_{33} & Tr_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (3)$$

where Ro_{ij} are the components of the rotation matrix from the camera coordinate system to the real-world coordinate system

and Tr_1, Tr_2, and Tr_3 are the components of the translation matrix from the camera coordinate system to the real-world coordinate system. The transformation result oX is evidenced in (4):

$$\begin{Bmatrix} x' = & Ro_{11}x + & Ro_{12}y + & Ro_{13}z + & Tr_1 \\ y' = & Ro_{21}x + & Ro_{22}y + & Ro_{23}z + & Tr_2 \\ z' = & Ro_{31}x + & Ro_{32}y + & Ro_{33}z + & Tr_3 \end{Bmatrix} \quad (4)$$

From the coordinates of the four points of point cloud data in the camera's coordinate system, their coordinates in the matrix are defined as:

$$\begin{bmatrix} 1 & z_1 & y_1 & x_1 \\ 1 & z_2 & y_2 & x_2 \\ 1 & z_3 & y_3 & x_3 \\ 1 & z_4 & y_4 & x_4 \end{bmatrix} \quad (5)$$

The transformation matrix according to the *x*, *y*, *z* axes is presented by:

$$\theta_1 = \begin{bmatrix} Tr_1 \\ Ro_{13} \\ Ro_{12} \\ Ro_{11} \end{bmatrix}; \theta_2 = \begin{bmatrix} Tr_2 \\ Ro_{23} \\ Ro_{22} \\ Ro_{21} \end{bmatrix}; \theta_2 = \begin{bmatrix} Tr_3 \\ Ro_{33} \\ Ro_{32} \\ Ro_{31} \end{bmatrix} \quad (6)$$

The results of the transformation are:

$$X' = \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \\ x'_4 \end{bmatrix}; Y' = \begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix}; Z' = \begin{bmatrix} z'_1 \\ z'_2 \\ z'_3 \\ z'_4 \end{bmatrix} \quad (7)$$

where $x'_1, x'_2, x'_3, x'_4, y'_1, y'_2, y'_3, y'_4, z'_1, z'_2, z'_3, z'_4$ are the ordinates of four points of the point cloud data in the real-world coordinate system. From this, a linear equation presented as formula in (8) is derived:

$$\begin{aligned} X' &= A\ x\ \theta_1 \\ Y' &= Ax\theta_2 \\ Z' &= A\ x\ \theta_3 \end{aligned} \quad (8)$$

To estimate $\theta_1, \theta_2, \theta_3$, the Least Squares method is used [21]:

$$\begin{aligned} \theta_1 &= A^T A^{-1} A^T X' \\ \theta_2 &= A^T A^{-1} A^T Y' \\ \theta_3 &= A^T A^{-1} A^T Z' \end{aligned} \quad (9)$$

Finally, the conversion matrix between the camera coordinate system and the real-world coordinate system is of the form $(\theta_1; \theta_2; \theta_3)$. The coordinates of the center in the real-world coordinate system are calculated by:

$$\begin{aligned} x_c &= \frac{x'_1 + x'_2 + x'_3 + x'_4}{4} \\ y_c &= \frac{y'_1 + y'_2 + y'_3 + y'_4}{4} \\ z_c &= \frac{z'_1 + z'_2 + z'_3 + z'_4}{4} \end{aligned} \quad (10)$$

The ground-truth data results of the motion trajectory in the real-world coordinate system are exhibited in Figure 5.

When collecting data and preparing the ground-truth data of the TQU-SLAM benchmark dataset, the following difficulties were encountered: (a) When collecting color and depth image data, many color image frames were blurred due to shaking during movement. Therefore, when the first recording was completed, the blurred frames on the color image were removed and were correspondingly removed on the depth image. (b) There is a gap between the two types of cameras on the Intel RealSense D435 (color camera, depth camera). To bring these two types of data to the same center, it is necessary to perform the calibration process (find image displacement matrix). However, due to the lack of facilities, matrix search was performed and it was tested on the collected data. Finally, the matrix noticed in (1) was acquired. (c) In the process of building 3D point cloud data of four points as in (2), the problem that on the depth image there are many areas with missing data due to absorption of infrared rays from the sensor was encountered. To fill in the data of missing regions, one can perform the fill-holing process (was not performed in this study). With four points marked on the marker, if any point had a depth value of 0, the average depth value of its 99 neighboring points was taken as the depth value of that point.

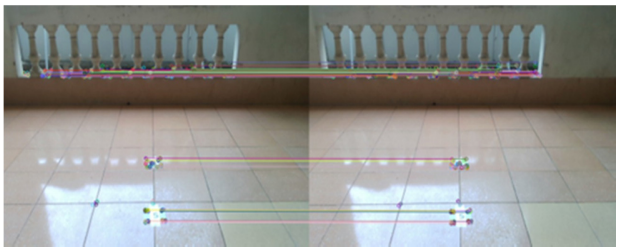

Fig. 5. ORB feature matching of two consecutive frames from the TQU-SLAM benchmark dataset.

III. COMPARATIVE STUDY OF BUILDING VISUAL ODOMETRY BASED ON KNOWLEDGE-BASED METHODS

As shown in Figure 1, the framework for constructing VO based on the knowledge-based methods from the collected images of the monocular camera includes five steps. Some recently published VO estimation techniques are presented.

A. PySLAM

PySLAM framework [21, 27], just like the knowledge-based methods, performs 6-DOF VO estimation with several steps. The feature extraction step, or keypoint detection, is the process of detecting features/keypoints between two consecutive frames, the features can be edges, corners, or blobs. Typically, the features are detected and extracted according to a feature descriptor, such as SHI_TOMASI, SIFT, SURF, ORB, AKAZE, KAZE [22], and BRISK. At the same time, feature extraction through DL networks such as VGG [25] and D2-Net [28] is also performed. Figure 6 portrays the match between the corresponding ORB features in two consecutive frames of the TQU-SLAM benchmark dataset.

In the motion estimation step, features are extracted from the monitored frame sequence from which the camera motion is estimated through the transformation matrix in (11):

$$T_{i,j-1}\begin{bmatrix} Ro_{i,j-1} & t_{i,j-1} \\ 0 & 1 \end{bmatrix} \quad (11)$$

where $Ro_{i,i-1}$ is the rotation matrix between two consecutive frames i and $i-1$ and $t_{i,i-1}$ is the translation vector between two consecutive frames i and $i-1$.

The spatial correlation matrix between frames i and $i-1$ is calculated based on (12) and the motion trajectory in 3D space is also calculated. A scale factor α is the shift factor between two consecutive frames.

$$E = \alpha * \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & -t_x & 0 \end{bmatrix} * Ro \quad (12)$$

where t is a translation matrix in the x, y, z direction. The calculation of the essential matrix is done in the epipolar plane system and by using epipolar constraints. This problem can be solved with the five-point algorithm or 5-point RANSAC [30, 31]. The RANSAC algorithm deployed here estimates the correspondence between two sets of keypoints. The points converted from the input set within a certain limit are called inliers. The algorithm will iterate about K times (K is calculated with (13)) [31] and output the largest number of inliers.

$$K = \frac{\log 1-p}{\log 1-w^s} \quad (13)$$

where p is the probability of finding a descriptor key-point, $s=2$ is the minimal number of samples needed to estimate a line model, and w is the likelihood ratio being inliers. This issue is summarized as follows.

In the local optimization step, camera pose estimation and motion trajectory errors are accumulated from camera pose estimation and transformation. Currently, there are two methods commonly applied to optimize the camera's movement trajectory in the environment. Firstly, the entire motion trajectory and camera pose are rechecked to minimize errors. The second is to use the Kalman algorithm or a particle filter to calibrate the map during data collection and edit the estimated camera pose when detecting new keypoints.

This framework works mainly depending on the feature extractor. The feature extractor's task is to calculate the input image according to the feature descriptor on two consecutive frames. Then, it performs feature matching and finds corresponding features on the two frames. From there, the state of motion can be estimated. At the same time, based on the displacement matrix between two feature sets, it estimates the motion trajectory. In PySLAM, no optimization is utilized to obtain the best motion trajectory and it is not suitable for real-time VO construction to meet the requirements of a real system. In particular, it does not use loop closure in VO construction. PySLAM provides a flexible architecture that allows the interchange ability of detectors and feature descriptors (SHI_TOMASI, SIFT, SURF, ORB, AKAZE, KAZE [22], and BRISK). Various feature matching techniques such as FLANN or BRUTE can be deployed for tracking as well as optical flow methods. For example, SHI_TOMASI is a detector similar to Harris, but more efficiently computes a corner by only accepting it if the R value is greater than the specified threshold value. This function allows setting the quality level and minimum detection distance. The quality level ranges from 0 to 1 and specifies an upper limit to remove corners. The default quality level is usually 0.01 in which, optical flow and Shi-Tomasi corner detection are used to identify features and track them between pairs of frames in what is called Lucas-Kanade-Tomasi tracking. Motion is obtained from spatial and temporal variations in image brightness, i.e. image gradients.

B. DPVO (Deep Patch Visual Odometry)

DPVO [32] is a DL framework that combines a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN). With the input image data being an RGB image, DPVO performs per pixel feature extraction employing ResNet and uses it to calculate similarities between images in the frame sequence. Next, two residual networks are utilized, the first network is used to extract matching features, and the second is deployed to extract contextual features. The first layer of each network performs convolution with a size of 7×7, the next two residual blocks have a size of 32×32, and the next two residual blocks have a size of 64×64, as illustrated in Figure 6. Finally, the output of the feature vector is the ¼ of the size of the input image. Next, a two-level feature pyramid is constructed with sizes 1/4 and 1/8, respectively, with the resolution of the input image based on a 4×4 filter with average pooling to the matching features to estimate the amount of optical flow. Then, p×p patches are obtained from the feature map with random positions in image space (pixel space) following the bilinear sampling technique. The patches map is created by linking patches and it has the same size as the feature map on pixels. DPVO used a patch graph to represent the relationship between patches and video frames. The projections of patches on the frames are the patches' movement trajectories. At the same time, DPVO also proposed an update operator that is a RNN that iteratively fine-tunes the depth of each patch and the camera pose of each frame in the video.

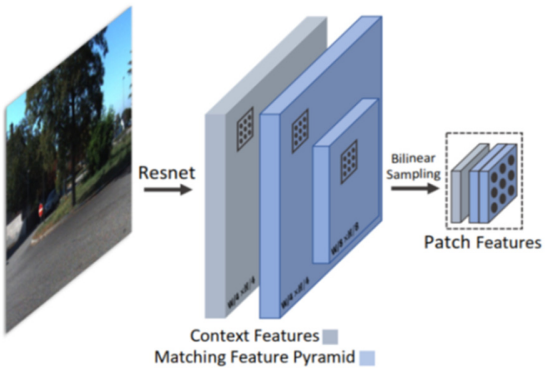

Fig. 6. Illustration of the DPVO architecture.

IV. EXPERIMENTAL RESULTS

A. Evaluation Measure

To evaluate the results of the VO algorithm based on the feature-based methods, the trajectory error (Err_d) was calculated, being the distance error between the ground truth AT_i and the estimated motion AT_i trajectory. Err_d is calculated according to:

$$Err_d = \frac{1}{N}\sqrt{\|AT_i - AT_i\|^2} \quad (14)$$

B. Evaluation Parameters

The PySLAM framework [20], with all features integrated was used. PySLAM development source code is Python v3.9 language and programmed on Ubuntu 20.04. At the same time, there is support from several open-source libraries, such as Numpy (1.18.2), OpenCV (4.5.1), PyTorch (>= 1.4.0), and Tensorflow-gpu (1.14.0). PySLAM framework was implemented on a computer with the following configuration: CPU i5 12400f, 16G DDr4, GPU RTX 3060 12GB.

C. Results and Discussion

The results of estimating VO/camera pose/camera trajectory on the TQU-SLAM benchmark dataset when using SHI_TOMASI, SIFT, SURF, ORB, ORB2, AKAZE, KAZE [18], BRISK, and VGG to extract features are portrayed in Table II. The average distance error of the SHI_TOMASI feature is the lowest (Err_d= 7.29 mm). The ratio of detected and extracted frames characterized for 6-DOF camera pose estimation is observed in Table III. The average ratio of the detected frames with the SHI_TOMASI feature is the highest (r_d = 98.97%).

Figure 8 presents the results of estimating VO of FO-D of the TQU-SLAM benchmark dataset when using the SHI_TOMASI features extracted from RGB images. Figure 8 also demonstrates that SHI_TOMASI features are extracted better, more in 1st - FO-D compared to other times.

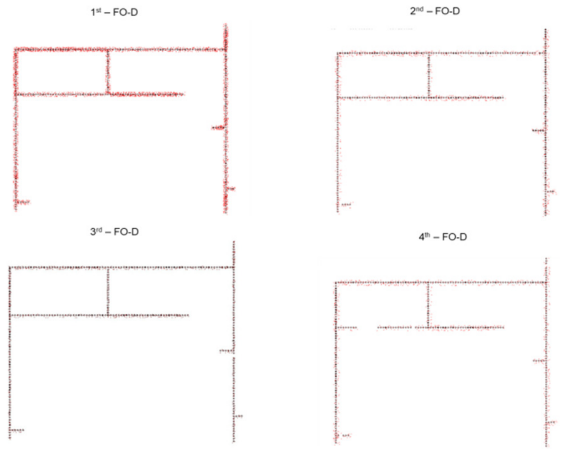

Fig. 7. Illustration of the results of estimating VO of TQU-SLAM benchmark dataset when using the SHI_TOMASI features. The results are presented on FO-D data. The black points belong to the ground truth trajectory, the red points are the estimated camera position.

TABLE II. RESULTS OF ESTIMATING VO ON THE TQU-SLAM BENCHMARK DATASET WHEN USING THE EXTRACTED FEATURES SHI_TOMASI, SIFT, SURF, ORB, ORB2, AKAZE, KAZE, BRISK, AND VGG

Dataset/Methods	**Features**	**Distance Error (Err_d(mm)) of the TQU-SLAM benchmark dataset**							
		1st		2nd		3rd		4th	
		FO-D	OP-D	FO-D	OP-D	FO-D	OP-D	FO-D	OP-D
PySLAM	SHI-TOMASI	6,019	2.903	5.938	6.836	14.42	10.978	8.224	3.050
	SIFT	38.93	13.02	37.63	13.58	23.55	1.63	32.02	2.11
	SURF	39.26	13.59	38.07	13.57	27.73	38.81	38.98	14.29
	ORB	1.42	14.67	32.48	12.19	25.93	13.29	0.75	2.07
	ORB2	8.97	9.14	8.19	2.88	20.33	12.78	10.54	4.32
	AKAZE	40.34	6.72	37.75	12.67	30.60	6.78	37.15	2.11
	KAZE	38.32	15.55	31.30	12.45	15.89	30.87	41.66	17.22
	BRISK	1.62	14.38	28.87	13.56	12.43	1.67	9.54	2.16
	VGG16	11.67	11.66	11.66	11.67	0.65	11.66	0.75	2.11

TABLE III. RATIO OF FRAMES WITH DETECTED FEATURES/KEYPOINTS ON THE TQU-SLAM BENCHMARK DATASET WHEN USING SHI_TOMASI, SIFT, SURF, ORB, ORB2, AKAZE, KAZE, BRISK, AND VGG

Dataset/Methods	**Features**	**Aspect ratio fails to detect keypoints of TQU-SLAM benchmark dataset (r_d(%))**							
		1st		2nd		3rd		4th	
		FO-D	OP-D	FO-D	OP-D	FO-D	OP-D	FO-D	OP-D
PySLAM	SHI_TOMASI	99.17	99.02	99.72	99.85	98.97	98.97	97.96	98.09
	SIFT	99.69	79.57	97.18	76.65	44.50	42.58	86.63	51.80
	SURF	99.67	97.84	95.51	94.66	95.76	98.95	96.76	97.66
	ORB	2.84	34.84	45.39	49.78	7.02	38.48	1.53	3.04
	ORB2	98.97	99.23	98.48	99.28	95.64	97.95	94.87	97.02
	AKAZE	88.66	9.71	76.42	34.20	67.65	8.94	54.61	3.08
	KAZE	90.36	25.51	75.83	46.15	89.29	9.04	76.36	19.20
	BRISK	1.82	16.26	6.25	9.32	39.81	28.98	12.18	3.10
	VGG	6.88	45.02	36.90	40.26	45.92	39.39	1.35	17.86

TABLE IV. RATIO OF FRAMES WITH DETECTED FEATURES/KEYPOINTS ON THE TQU-SLAM BENCHMARK DATASET WHEN USING DPVO TO EXTRACT FEATURES

Dataset/Methods	**Experimental**	**Aspect ratio fails to detect keypoints of TQU-SLAM benchmark dataset (r_d(%))**							
		1st		2nd		3rd		4th	
		FO-D	OP-D	FO-D	OP-D	FO-D	OP-D	FO-D	OP-D
DPVO	Average	49.85	49.44	50.00	50.00	50.00	50.00	50.90	47.23

Figure 9 exhibits the results of estimating points on the moving trajectory on 1st-FO-D of the TQU-SLAM benchmark dataset when utilizing the features extracted from VGG. The ratio of the number of detected and extracted frames characterized to estimate the VO of the DPVO is shown in Table IV. The rate is low, only 49.68%. The results in Figure 9 also indicate that the features extracted are limited when employing VGG. There are many frames whose features cannot be extracted. Therefore, the camera's position cannot be estimated.

Figure 10 showcases the difficult feature extraction on frame pairs when using VGG.

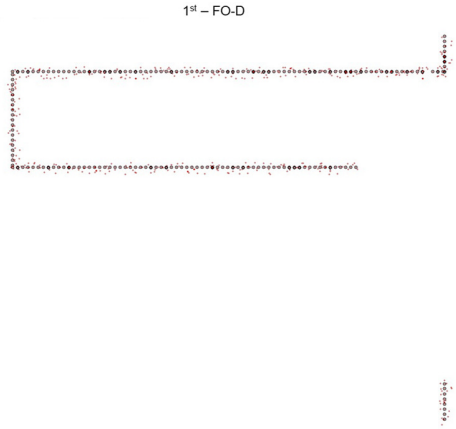

Fig. 8. Illustration of the results of estimating VO of 1st-FO-D of TQU-SLAM benchmark dataset when using the features extracted by VGG. The black points belong to the ground truth trajectory, the red points are the estimated camera position.

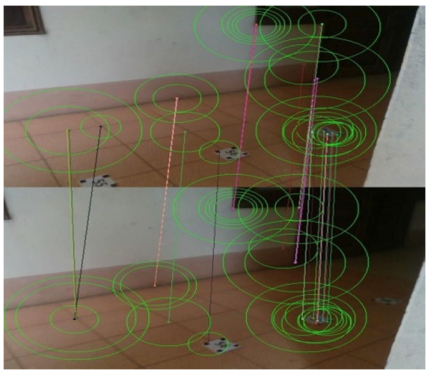

Fig. 9. Illustration of the feature matching extracted from the VGG of frame pair of 1st-FO-D of the TQU-SLAM benchmark dataset.

V. CHALLENGES

As shown in Figures 7-10 and Tables III and IV, the TQU-SLAM benchmark dataset contains some challenges for VO construction. Firstly, regarding lighting conditions, the obtained RGB images have very weak lighting conditions, and the light intensity is uneven. There are road sections with adequate lighting, but there are road sections where additional light from a mobile phone has to be used to assist. Therefore, the keypoints between two consecutive frames are not detected. Second, the collected environment is highly homogeneous. The image data obtained by the Intel RealSense D435 are mainly floor data with a small part of wall and railing data. Therefore, the data do not have great discrimination between frames, in the scene there are few objects in the environment. Although markers have been posted on the floor, the number of keypoints detected on the two frames is not large. This leads to failure to estimate the 6-DOF camera pose across multiple frames.

VI. CONCLUSIONS

VO systems are widely applied in pathfinding robots, autonomous vehicles operating in industry, etc., and DL has had impressive results in building Visual SLAM and VO systems. However, DL requires a large amount of data to train the model. This paper introduces the TQU-SLAM benchmark dataset, collected from an RGB- D image sensor (Intel RealSense D435) moving in the corridors of three buildings with FO-D = 230.63 m and OP-D = 228.13 m. The ground-truth data of the TQU-SLAM benchmark dataset include 6-DOF camera pose/ camera trajectory and a 3D point cloud. It was also tested to estimate VO utilizing some traditional features and features extracted from DL such as VGG based on the PySLAM framework. At the same time, tests were performed on a pre-trained model trained from the KIti database of a DL-based method like DPVO. Among them, the SHI_TOMASI features gave the best results (Err_d = 7.29 mm, r_d = 98.97%). The results based on DPVO have a rate of frames with no feature extraction between frame pairs up to 50%. This proves that the TQU-SLAM benchmark dataset is very challenging to build VO from monocular RGB-D cameras. In the future, research will be conducted on point cloud data in computer vision and DL [33-35]. Further comparative studies will be additionally performed on the use of DL models to build VO, such as fine-tuning models and pre-trained models to estimate VO on the TQU-SLAM benchmark dataset.

ACKNOWLEDGMENT

This research is funded by Tan Trao University in TuyenQuang, Vietnam.

REFERENCES

[1] K. Wang, S. Ma, J. Chen, F. Ren, and J. Lu, "Approaches, Challenges, and Applications for Deep Visual Odometry: Toward Complicated and Emerging Areas," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 1, pp. 35–49, Mar. 2022, https://doi.org/10.1109/TCDS.2020.3038898.

[2] A. Neyestani, F. Picariello, A. Basiri, P. Daponte, and L. D. Vito, "Survey and Research Challenges in Monocular Visual Odometry," in *International Workshop on Metrology for Living Environment (MetroLivEnv)*, Milano, Italy, Dec. 2023, pp. 107–112, https://doi.org/10.1109/MetroLivEnv56897.2023.10164057.

[3] L. R. Agostinho, N. M. Ricardo, M. I. Pereira, A. Hiolle, and A. M. Pinto, "A Practical Survey on Visual Odometry for Autonomous Driving in Challenging Scenarios and Conditions," *IEEE Access*, vol. 10, pp. 72182–72205, Jan. 2022, https://doi.org/10.1109/ACCESS.2022.3188990.

[4] M. A. Haq, A. K. Jilani, and P. Prabu, "Deep learning based modeling of groundwater storage change," *Computers, Materials and Continua*, vol. 70, no. 3, pp. 4599–4617, 2022, https://doi.org/10.32604/cmc.2022.020495.

[5] M. A. Haq, "CDLSTM: A novel model for climate change forecasting," *Computers, Materials and Continua*, vol. 71, no. 2, pp. 2363–2381, 2022, https://doi.org/10.32604/cmc.2022.023059.

[6] M. A. Haq, "Smotednn: A novel model for air pollution forecasting and aqi classification," *Computers, Materials and Continua*, vol. 71, no. 1, pp. 1403–1425, 2022, https://doi.org/10.32604/cmc.2022.021968.

[7] M. A. Haq, "Planetscope Nanosatellites Image Classification Using Machine Learning," *Computer Systems Science and Engineering*, vol. 42, no. 3, pp. 1031–1046, Feb. 2022, https://doi.org/10.32604/csse.2022.023221.

[8] M. A. Haq, "CNN Based Automated Weed Detection System Using UAV Imagery," *Computer Systems Science and Engineering*, vol. 42, no. 2, pp. 837–849, Jan. 2022, https://doi.org/10.32604/csse.2022.023016.

[9] M. A. Haq, G. Rahaman, P. Baral, and A. Ghosh, "Deep Learning Based Supervised Image Classification Using UAV Images for Forest Areas Classification," *Journal of the Indian Society of Remote Sensing*, vol. 49, no. 3, pp. 601–606, Mar. 2021, https://doi.org/10.1007/s12524-020-01231-3.

[10] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004, https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[11] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," in *European Conference on Computer Vision*, Graz, Austria, Dec. 2006, pp. 404–417, https://doi.org/10.1007/11744023_32.

[12] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *International Conference on Computer Vision*, Barcelona, Spain, Nov. 2011, pp. 2564–2571, https://doi.org/10.1109/ICCV.2011.6126544.

[13] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary Robust invariant scalable keypoints," in *International Conference on Computer Vision*, Barcelona, Spain, Nov. 2011, pp. 2548–2555, https://doi.org/10.1109/ICCV.2011.6126542.

[14] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *7th International Joint Conference on Artificial Intelligence*, Vancouver, BC, Canada, Aug. 1981, vol. 2, pp. 674–679.

[15] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, Jun. 2012, pp. 3354–3361, https://doi.org/10.1109/CVPR.2012.6248074.

[16] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013, https://doi.org/10.1177/0278364913491297.

[17] M. Menze and A. Geiger, "Object Scene Flow for Autonomous Vehicles," in *IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, Jun. 2015, pp. 3061–3070.

[18] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal, Oct. 2012, pp. 573–580, https://doi.org/10.1109/IROS.2012.6385773.

[19] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *European Conference on Computer Vision*, Florence, Italy, Oct. 2012, pp. 746–760, https://doi.org/10.1007/978-3-642-33715-4_54.

[20] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE International Conference on Robotics and Automation*, Hong Kong, China, Jun. 2014, pp. 1524–1531, https://doi.org/10.1109/ICRA.2014.6907054.

[21] L. M. Hodne, E. Leikvoll, M. Yip, A. L. Teigen, A. Stahl, and R. Mester, "Detecting and Suppressing Marine Snow for Underwater Visual SLAM," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, LA, USA, Jun. 2022, pp. 5101–5109.

[22] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 5, no. 31, pp. 1147–1163, 2015, https://doi.org/10.1109/TRO.2015.2463671.

[23] 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, https://doi.org/10.1109/TRO.2017.2705103.

[24] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "KAZE Features," in *European Conference on Computer Vision*, Florence, Italy, Oct. 2012, pp. 214–227, https://doi.org/10.1007/978-3-642-33783-3_16.

[25] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition." arXiv, Apr. 10, 2015, https://doi.org/10.48550/arXiv.1409.1556.

[26] V. H. Le, "DATA_TQU_SLAM." [Online]. Available: https://drive.google.com/drive/folders/16Dx_nORUvUHFg2BU9mm8aBYMvtAzE9m7.

[27] L. Freda, "luigifreda/pyslam." Jun. 03, 2024, [Online]. Available: https://github.com/luigifreda/pyslam.

[28] "Bài 3: Linear Regression," *Machine Learning cơ bản*. https://machinelearningcoban.com/2016/12/28/linearregression/.

[29] M. Dusmanu *et al.*, "D2-Net: A Trainable CNN for Joint Detection and Description of Local Features," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, Jun. 2019. 9

[30] F. Fraundorfer and D. Scaramuzza, "Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 78–90, Jun. 2012, https://doi.org/10.1109/MRA.2012.2182810.

[31] V.-H. Le, H. Vu, T. T. Nguyen, T.-L. Le, and T.-H. Tran, "Acquiring qualified samples for RANSAC using geometrical constraints," *Pattern Recognition Letters*, vol. 102, pp. 58–66, Jan. 2018, https://doi.org/10.1016/j.patrec.2017.12.012.

[32] Z. Teed, L. Lipson, and J. Deng, "Deep Patch Visual Odometry," *Advances in Neural Information Processing Systems*, vol. 36, pp. 39033–39051, Dec. 2023.

[33] A. Alayed, R. Alidrisi, E. Feras, S. Aboukozzana, and A. Alomayri, "Real-Time Inspection of Fire Safety Equipment using Computer Vision and Deep Learning," *Engineering, Technology & Applied Science Research*, vol. 14, no. 2, pp. 13290–13298, Apr. 2024, https://doi.org/10.48084/etasr.6753.

[34] A. N. Sazaly, M. F. M. Ariff, and A. F. Razali, "3D Indoor Crime Scene Reconstruction from Micro UAV Photogrammetry Technique," *Engineering, Technology & Applied Science Research*, vol. 13, no. 6, pp. 12020–12025, Dec. 2023, https://doi.org/10.48084/etasr.6260.

[35] M. Ramzan, M. S. Farooq, A. Zamir, W. Akhtar, M. Ilyas, and H. U. Khan, "An Analysis of Issues for Adoption of Cloud Computing in Telecom Industries," *Engineering, Technology & Applied Science Research*, vol. 8, no. 4, pp. 3157–3161, Aug. 2018, https://doi.org/10.48084/etasr.2101.