**1. M. I. Idris[1], 2. H. Hussin[1], 3. S. A. M. Junos[1], 4. W. H. M. Saad[1], 5. M. N. Shah Zainudin[1], 6. N. M. Yatim[1], 7. A. S. Jaafar[1], 8. M. R. Kamarudin[1], 9. K. M. Saipullah[2]**

[1]Fakulti Kejuruteraan Elektronik dan Kejuruteraan Komputer (FKEKK), Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia.
[2]Autonomous Robotics Division, Strukture Robotics Sdn. Bhd

doi:10.15199/48.2022.09.08

# VSLAM analysis using various ORBSLAM parameters setting

***Abstract**: SLAM or simultaneous localization and mapping system, is a system that determines the orientation and position of a robot by creating a detailed map of the environment while simultaneously tracking where the robot is within the environment. This project aims to use the camera as SLAM primary sensor to replace the LiDAR sensor frequently used in autonomous robots. ORBSLAM is used as the main algorithm, and a few settings are being adjusted to get the most accurate results. The Absolute Trajectory Error and Relative Pose Error are used to evaluate the algorithm's accuracy. After the most optimized setting is found, the setting is used for real-time mapping in an unknown environment.*

*Streszczenie. SLAM lub system jednoczesnej lokalizacji i mapowania, to system, który określa orientację i pozycję robota, tworząc szczegółową mapę otoczenia, jednocześnie śledząc, gdzie robot znajduje się w środowisku. Celem tego projektu jest wykorzystanie kamery jako głównego czujnika SLAM, który zastąpi czujnik LiDAR często używany w autonomicznych robotach. ORBSLAM jest używany jako główny algorytm, a kilka ustawień jest dostosowywanych, aby uzyskać jak najdokładniejsze wyniki. Bezwzględny błąd trajektorii i względny błąd pozycji służą do oceny dokładności algorytmu. Po znalezieniu najbardziej zoptymalizowanego ustawienia jest ono używane do mapowania w czasie rzeczywistym w nieznanym środowisku.* (***Analiza VSLAM przy użyciu różnych ustawień parametrów ORBSLAM***)

**Keywords:** VSLAM; ORBSLAM; Absolute Trajectory Error (ATE); Pose Relative Error (RPE).
**Słowa kluczowe**: VSLAM, ORBSLAM, błąd trajektorii

## Introduction

Nowadays, many inventions are being invented to reduce the burden of labour on humankind. Invention such as robot is being used to replace human as labour to do challenging jobs. Nevertheless, there is a need for a particular thing: machines do physical tasks and think and make decisions. An autonomous robot that can travel freely in a static or dynamic environment is required to reach all the stated objectives. The perfect navigation system must be built so that the robot always knows where should it go [1].

In today's world, the main localization system used is the Global Navigation Satellite System (GNSS), which has been fantastic in providing an absolute positioning on Earth's surface [2]. Nevertheless, the GNSS system is not always perfect if the environment is uneven. Furthermore, some unexpected geographical conditions can lead to errors of some meters, which is not acceptable for safe robotic autonomous navigation. Moreover, a mobile robot needs to be able to navigate even in a dynamic environment with potential obstacles and does not always have any prior information about its environment

Smooth and safe mobile robot navigation through the cluttered environment from the start position to the goal position by following the safe path and producing optimal path length is the main aim of mobile robot navigation. Several techniques have been explored by researchers for robot navigation path planning regarding this matter. Artificial intelligence plays an important role to make things intelligent to achieve this target. Path planning has been considered the most common problem for robot navigation, and robots have to move from starting position to the goal position by avoiding obstacles [3].

## Literature Review

Autonomous navigation for mobile robots has been a very active research area in the past few decades. A navigation system is an essential feature in robotics in order to develop an autonomous robot. SLAM is being introduced in the robotic industry to get more precise and accurate navigation. SLAM is a process in which a robotic system constructs a map of the environment using any devices attached while simultaneously estimating its position in the environment [4]. In order to gather information about the environment, the current position of the robot must be known. Therefore, the robot's position must always be localized in the incomplete map to get more information about the environment.

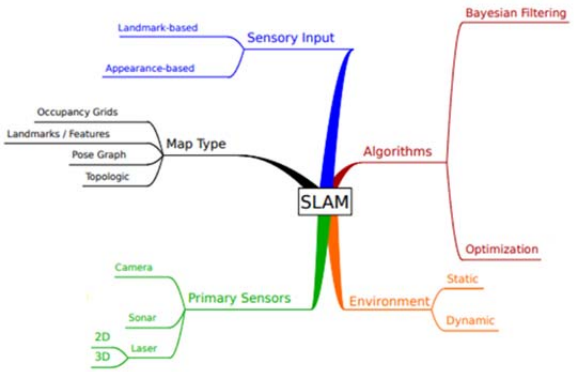


Figure 1: A taxonomy of the SLAM problem [4]

Figure 1 shows the taxonomy of the SLAM problem that being occurred. Precise and accurate localization is the most crucial part of the SLAM process. Therefore, an inaccurate localization produces an error map and makes future localization more difficult. SLAM is a very hard process because the robot has to adapt and face problems such as the noise from the sensors and the inertia effect of the motor used [4]. The SLAM problem is formulated in the Bayes Net diagram shown in Figure 2.

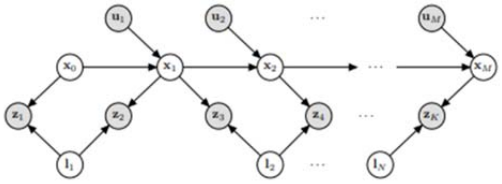

Figure 2: Bayes Net Diagram [4]

$X = \{x_i\}$, $i \in 0\ldots, M$ in Figure 2 is being defined as the robot's trajectory through time, with xi as the vector representation of the state or pose of the robot [4]. The

state of a robot, such as velocity and acceleration, is being parameterized in SE (2). SE (2) is a coordinate system for the robot's configuration space [5]. The SE (2) is described as a planar rigid body motion represented by the Special Euclidean group in two dimensions. The SE (2) system point out where the robot should be going on the planar coordinate so that the robot processes the correct plan that it should be heading [4]. Meanwhile, L = {$l_j$}, j ∈ {1, . . ., N} is the latent position of the landmark and Z = {zk}, k ∈ {1, . . ., K} is the noisy measurements to the landmarks detected. The robot motion problems due to the noise or inertia that been recovered is denoted by U = {$u_i$}, i = {1, . . ., M}[4].

i. Visual SLAM

Visual SLAM, also known as VSLAM, is one of the SLAM approaches that has received much attention from the robotics field because of its ability to overcome all other sensor shortcomings [5]. VSLAM has been researched more frequently in recent years. The camera is a detector for this technology that collects more data, has important object identification functions, senses at a much better resolution than radar, is less expensive, is easier to transport, and operates more simply. Given all of the advantages, it's no wonder that many institutions and researchers are focusing more on this strategy [5]. VSLAM technology has gradually progressed from laboratory research to practical applications. A few popular types of camera sensors are frequently used, such as monocular cameras, binocular cameras, and RGB-D depth cameras, in terms of sensing[6].

Figure 3 shows the technical framework of the VSLAM that is being applied to the VSLAM system [6]. The framework is the classic framework for the SLAM. The VSLAM framework consists of four primary modules, and each of them consists of many algorithms: Visual-inertial odometry (VIO), Optimization, Loop Closing, and Mapping [6].

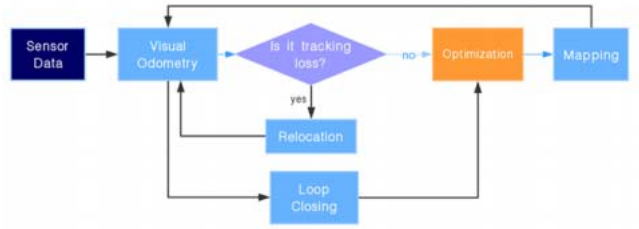


Figure 3: Technical Framework of VSLAM

ii. Comparison of the VSLAM Algorithms

Numerous VSLAM algorithms would be suitable to be implemented for this project. However, each method has advantages and disadvantages, limiting its applicability to specific environments. Table 1 summarizes all of the VSLAM algorithms developed over the years.

Each of the algorithms shown in Table 1 has its benefits and drawbacks. The area determines the compatibility of most algorithms they aim to cover, and some of them contain a loop closure feature that reduces error in the process. Because of its cost flexibility and the number of open resources available, the ORB-SLAM appears to outperform all other methods in robot navigation usage [7].

**Methodology**

i. Working Principle

The VSLAM for the autonomous mapping project is implemented into the Jetson Nano (B01) as the central processor and core [8]. The working principle of this project as the algorithm chosen is tested with a few parameter settings to find the most optimized setting that can be used for real-time mapping.

A dataset of Euroc is being used to differentiate the result and make the analysis comparable for each parameter setting. After that, the Absolute Trajectory Error (ATE) evaluation and Relative Pose Error (RPE) evaluation are used to see the value of error that is being received in each of the settings [9]. The setting with the least RPE and ATE error is chosen for the real-time mapping.

Then, a robot hardware component is used to carry the camera and the Jetson Nano to operate the algorithm in a real-time environment. The robot is equipped with the L298N motor driver and 2 Dc motor to control the robot's movement.

Table 1: Summary of all the VSLAM algorithms[7]

| Algorithm | Method | Loop Closure | Cost | Disadvantage |
|---|---|---|---|---|
| Mono-SLAM | Feature Method | No | Flexible (based on the camera used) | Difficult to do the calculation if covering a large environment. |
| PTAM | Feature Method | No | Flexible (based on the camera used) | Lacking of loop closure feature, it's only suited for small-scale AR |
| LSD-SLAM | Direct Method | Yes | Rm100++ | Low quality of camera resulting lower quality results |
| ORB-SLAM | Feature Method | Yes | Flexible (based on the camera used) | Not suitable for tracking in high-density requirements |
| RGB-D SLAM | RGB-D method | Depend on algorithm | Rm300++ | Complicated to deploy |

ii. Project Overview

The Jetson Nano function sends the signal obtained from the user and sends the information gathered to the Arduino nano. The Arduino received the signal and started controlling the motor driver based on the given signals.

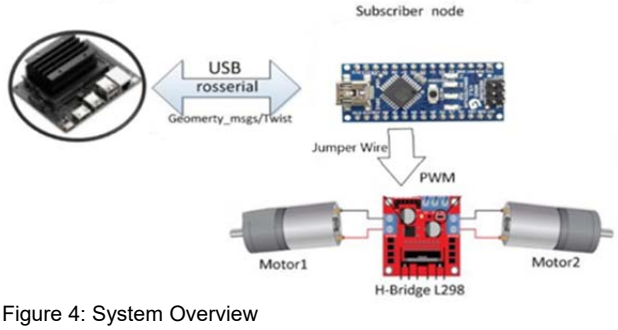


Figure 4: System Overview

iii. Communication with Arduino Nano

To make this project work as one single unit, the Arduino Nano received the Jetson Nano (B01) signal to control all the other components. A communication package called rosserial is used to communicate with both boards [10]. Rosserial is a protocol used to send data through a serial interface, and the integration between both boards is shown in Figure 5 [10].

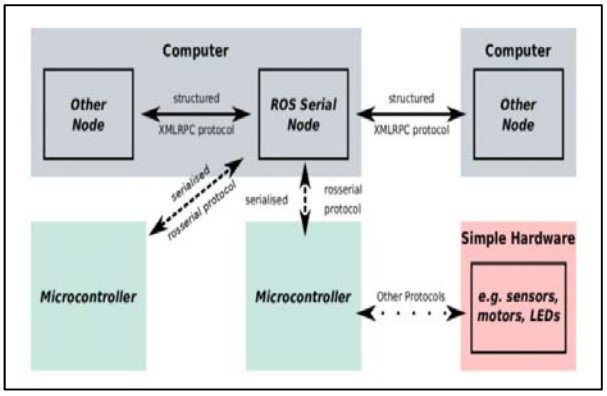

Figure 5: Rosserial Integration with Hardware Components

Based on Figure 5, Jetson Nano serves as the computer in this project, while the Arduino Nano serve as the microcontroller. In this implementation, rosserial-server is a publishing node, whereas rosserial-client is a subscriber node, albeit this can occasionally be reversed[11]. A micro-USB cable is utilized as a communication link between the Jetson Nano and the Arduino Nano for this communication protocol to be implemented[11].

iv. Algorithm Selection

To start processing the map, a correct VSLAM Algorithm has to be selected. A proper algorithm selection can make a major difference in the result. When it comes to getting the best outcome from the camera input without being considering the light intensity, the ORBSLAM algorithm is the best algorithm that can be found.

v. Euroc Dataset

Dataset is a sequence of images or videos in a certain environment used to test performance. In this project, Euroc dataset is being used. Euroc datasets are the dataset that is released by the Swiss Federal Institute of Technology Zurich [12]. The EuRoC MAV dataset is a visual-inertial dataset collected on a Micro Aerial Vehicle (MAV). The Euroc dataset already has a ground-truth value, making the evaluation much more manageable. The Euroc ground-truth value is being obtained using the Vicon motion capture system (6D pose), Leica MS50 laser tracker (3D position) and Leica MS50 3D structure scan [12].

In this project, the Euroc Dataset is the main performance benchmark for the ORBSLAM algorithm settings chosen to select the most optimized setting for real-time mapping.

vi. Camera Selection

The camera that is used in this project is the Logitech c270. The Logitech c270 is being chosen due to its capability in processing V4L2, a video interface for any Linux OS. The V4L2 makes the interaction much easier with the Jetson Nano(B01) to troubleshoot if there are any problems.

vii. Camera Calibration

Before beginning the real-time mapping using the VSLAM algorithm that is selected, firstly, the camera must be calibrated to get the right range of detection required[9]. Camera calibration is necessary to find the measuring distance from images acquired with a stereo camera, monocular camera, or processing images for object recognition.

The most used model for perspective camera assumes a pinhole projection system: the image is formed by the intersection of the light rays from the objects through the centre of the lens (projection centre), with the focal plane as shown in Figure 6 [9].

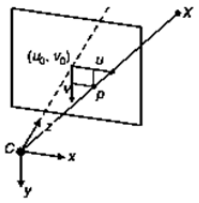

Figure 6: Perspectives Projection of Camera Through Object [9]

Based on Figure 6 perspectives projection, the X line on the figure can be X = [x,y,z]^T as a scene point in the camera reference frame, and p = [u,v]^T is the camera projection on the image plane that is measured in pixels [9]. The perspective projection equation gives the mapping from the 3-D world to the 2-D image, as shown in Equation 1[9].

$$(1)\ \boldsymbol{\lambda}\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{v}\\ \boldsymbol{1}\end{bmatrix} = \boldsymbol{KX} = \begin{bmatrix}\boldsymbol{\alpha_u} & \boldsymbol{0} & \boldsymbol{u_0}\\ \boldsymbol{0} & \boldsymbol{\alpha_v} & \boldsymbol{v_0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1}\end{bmatrix}\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{y}\\ \boldsymbol{z}\end{bmatrix}$$

where: λ =the depth factor for the camera, $\boldsymbol{\alpha}$=the focal length s of the camera used, $u_0$, $v_0$= the coordinates of the projection centre, x,y,z is equal to axis of the projection

Based on Equation 1, the parameter generated using the matrix equation is called the intrinsic parameter suitable for a camera with an angle of view 45° or less[9]. However, for the angle of more 45°, the effect of the radial distortion of the camera may become visible, and it can be modelled using the second- (or higher)-order polynomial[9].

*viii.* Evaluation

In this project, the Euroc dataset name V1_01 is used as a benchmark dataset to find the best accurate parameter setting before real-time mapping. The number of point cloud parameters on the ORBSLAM setting is being adjusted from the lowest possible value for the Monocular camera to the highest possible to see its effect on the performance and accuracy of the map generated using the datasets. In this section, all the evaluation methods to find the most accurate parameter for the ORBSLAM are explained.

ix. Relative Position Error (RPE)

The algorithm generates the estimated trajectory pose to evaluate the VSLAM algorithm performance and accuracy when finished processing. All the datasets' sequences are assumed as P_1,..,P_n ∈ SE(3). The trajectory pose of the ground truth is set as Q_1,..,Q_n∈ SE(3) [13]. Neither the estimated nor the ground truth trajectory, both sequences consist of homogeneous transformation matrices that represent the pose of the RGB optical frame from an (arbitrary) reference frame. The relative pose error (RPE) measures the local accuracy of the trajectory over a fixed time interval Δ of the sequences [13].

Therefore, in simple words, the relative pose error measures the drift of the pose trajectory based on the ground truth, which is very useful for evaluating visual odometry. In other words, it calculates the difference in relative motion between two poses which may be used to estimate drift. The relative pose error at time step i is defined as Equation 2.

$$(2)\ \mathbf{E_i} := (\boldsymbol{Q_i^{-1}}\,\boldsymbol{Q_i} + \boldsymbol{\Delta})(\boldsymbol{P_i^{-1}}\boldsymbol{P_i} + \boldsymbol{\Delta})$$

where: S= the rigid-body transformation, Q= the trajectory poses of the ground truth, P= the estimated trajectory poses that being generated by the algorithm, Δ= time interval value

Using a sequence of n camera poses, m = n − Δ is being obtained to find individual relative pose errors along the sequence [13]. The root mean squared error (RMSE) over the total time indices of the translational component is being calculated using:

$$(3) \quad \boldsymbol{RMSE}(\mathbf{E_{1:n}}, \boldsymbol{\Delta}) = \left(\frac{1}{m}\sum_{i=1}^{m}\left|\left|\boldsymbol{trans}(\boldsymbol{E_i})\right|\right|^2\right)^{\frac{1}{2}}$$

where: m=number of sequences, $\boldsymbol{trans}(\boldsymbol{E_i})$= transpose of absolute trajectory error at time step

Based on equation 3, the RMSE value should not be more than 1% due to the drift error being generated by the estimated pose. If the RMSE value exceeds 1%, the setting used for the processing cannot be used due to a higher pose relative error between the ground truth and the trajectory generated[13].

x. Absolute Pose Error

The most obvious accuracy of the algorithm can be evaluated by comparing the absolute distance between the estimated trajectory and the ground truth trajectory[13]. Both trajectories can be specified in arbitrary coordinate frames; however, both of them need to be aligned first to compare them. The method of Horn can be used to get both the trajectory aligned together, which determines the rigid-body transformation S that maps the predicted trajectory P1: n onto the ground truth trajectory Q1: n[13]. The absolute trajectory error at the time step may be determined using this transformation, as shown in Equation 4[13].

$$(4) \quad \boldsymbol{F_i} \coloneqq \boldsymbol{Q_i^{-1} S P_i}$$

where: S= the rigid-body transformation, Q= the trajectory poses of the ground truth, P= the estimated trajectory poses that being generated by the algorithm

Similarly, the ATE also evaluates the RMSE over all time indices of the translational components with the relative pose error. The RMSE formula can be defined as Equation 5 [13].

$$(5) \quad \boldsymbol{RMSE}(\mathbf{F_{1:n}}) \coloneqq \left(\frac{1}{n}\sum_{i=1}^{n}\left|\left|\boldsymbol{trans}(\boldsymbol{F_i})\right|\right|^2\right)^{\frac{1}{2}}$$

where: n=number of samples, $\boldsymbol{trans}(\boldsymbol{F_i})$= transpose of absolute trajectory error at time step

Compared to the RPE, ATE only consider the translational error while the RPE consider both the translational and considers both translational and rotational errors [13]. Therefore, the RPE is always slightly larger than the ATE.

**Result And Discussion**

i. Robot Control

A robotic device is being made for carrying all the important components to do the mapping process, obtaining the result of the real-time map using the selected algorithm. The user then controls the device using the teleop-twist keyboard in the ROS platform attached to the Jetson nano [14].

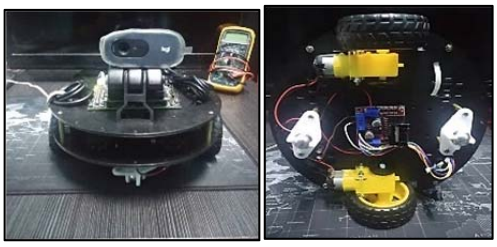

Figure 7: Result of the Robot components

The Arduino nano subscribes to the /cmd_vel topic and receives the signal based on what is being pressed on the keyboard and based on the signal sent; the motor moves as the instruction written on the nano coding [15].

Figure 8 shows the fully connected topic from the teleop-twist keyboard to the Arduino nano attached to the robot. The Arduino nano communicates to the Jetson nano using a micro-USB wire that is attached to the USB port of the Jetson nano[15]. The USB act as a power supply and, at the same time, gives the signal that is required for the Arduino to move the robot motor.

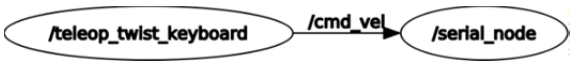


Figure 8: The ROS Subscriber Graph to Control the Robot

ii. ORBSLAM Selection

A few algorithms can be used for the VSLAM mapping process. Algorithms such as DSO and LSD SLAM are a few examples of the Monocular camera-based algorithm used in this project [7][16]. However, the ORBSLAM algorithm is chosen as the main algorithm for this project. The selection is influenced by its outstanding performance, which outperforms all the other algorithms in the Absolute Trajectory Error (ATE) evaluation based on the previous paper [9]. Table 2 show the ORBSLAM performances compared to other algorithms.

*Table 2: Comparison ORBSLAM with other Algorithm [16]*

| Algorithms | RMSE (m) | Mean (m) | Median (m) | Std (m) | Min (m) | Max (m) |
|---|---|---|---|---|---|---|
| LSD SLAM | 0.301 | 0.277 | 0.262 | 0.11 | 0.08 | 0.55 |
| ORB SL-AM | 0.166 | 0.159 | 0.164 | 0.04 | 0.04 | 0.25 |
| DSO | 0.459 | 0.403 | 0.419 | 0.21 | 0.00 | 0.76 |

Based on Table 2, the ORBSLAM Algorithm has the lowest trajectory drift error compared to the other two algorithms that are being selected. The error value of ORBSLAM from the RMSE, mean error, median error, Standard deviation, minimum error, and maximum error all has lower error value than the DSO and LSD SLAM algorithm [9]. In conclusion, the ORBSLAM has the lowest drift error for the table compared to other selected algorithms, producing the most accurate result. Therefore, the ORBSLAM is being used as the main algorithm tested in this project.

iii. Result Comparison

Figure 9 shows the result obtained using the 800,1000,1500, and 2000 Features per images parameter settings.

Based on Figure 9, the result shows all the maps developed using 800,1000,1500 and 2000 Features points per image parameter setup. The map generated using the 800 Features point per image shows the lowest essential graph connection (green link) between each keyframe (blue box); meanwhile, 1000 Features points per frame show more generated essential graph than 800 Features per image [17]. The environment map generated in the 1000 Features point per image also seems more accurate than the 800 Features per image setup. 1500 Features point per image and 2000 Features point per image, on the other hand, seems to have equal visualization of the environment that is being developed. However, the essential graph linked in the 2000 Features point per image is higher than the 1000 Features points per image set.

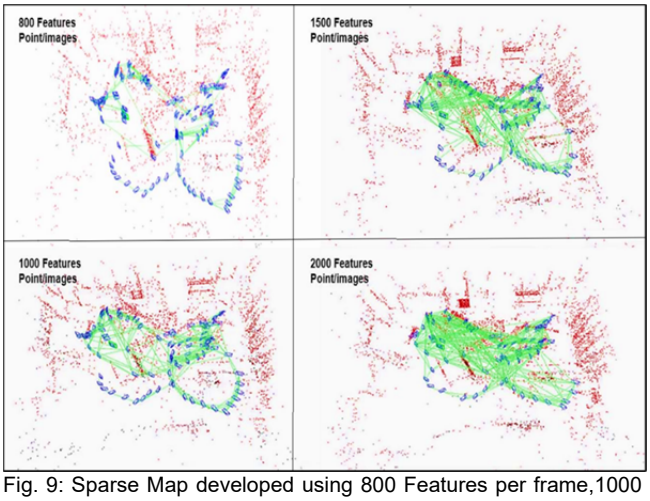


Fig. 9: Sparse Map developed using 800 Features per frame,1000 Features per frame, 1500 Features per frame and 2000 Features per frame. Keyframes (blue), Current Camera (green), MapPoint (black, red), Current Local MapPoint (red), Essential Graph (green) [17]

The higher essential graph connection means that the loop closing process in the map trajectory has more accuracy [17]. The 2000 Features points per image have the essential graph connection compared to the other parameter settings being tested. Therefore, theoretically, the 2000 Features per image should produce the most accurate map compared to the others[17].

*iv.* Evaluation

The ATE and RPE formula has been used and mentioned in Equation 3 and Equation 5. The final evaluation results in being plotted in Table 3.

Table 3: ORBSLAM performance in Euroc dataset

| Camera Parameter Adjustment | | | Average Total Matches Point | Mean Tracking Time(s) | ATE RMSE (m) | RPE RMSE (m) |
|---|---|---|---|---|---|---|
| a | b | c | | | | |
| 800 | 15 | 5 | 229 | 0.145 | 0.095464 | 0.674739 |
| 1000 | 20 | 7 | 253 | 0.172 | 0.098354 | 1.004954 |
| 1500 | 25 | 9 | 315 | 0.208 | 0.097415 | 0.928890 |
| 2000 | 30 | 11 | 432 | 0. 243 | 0.060098 | 0.932499 |

* a is Number of Features points per image, b is Features Point Value to Initialize and c is Min. Threshold Value for Features Point Detection.

Based on Table 3, the result shows that the 2000 Features point per image has the longest tracking time by 0.243 seconds compared to the closest parameter settings at 0.208 seconds. The high number of features point that it processes simultaneously makes the Jetson Nano (B01) GPU usage higher than usual and slower the overall process of the devices. In the ATE RMSE section, 2000 has the lowest error at 0.060098 compared to the other parameter. The rich information gathered through the highest features point detection results from the most accurate trajectory than the ground truth. However, considering the Pose error in the projection, the 800 Features point per image shows the slightest error compared to others by 0.674739.

v. Real-time mapping Result

Based on the evaluation result, the real-time mapping process has been done using the most suitable parameter based on the result obtained. The result is processed using the robot hardware made as a medium transporting the camera and the machine. However, the RMSE of the RPE and ATE evaluation for the real-time environment cannot be plotted due to the unavailability of the ground truth value for the real-time surroundings[18]. Figure 11 show the result of the mapping process in an unknown environment.

Fig. 10: The environment

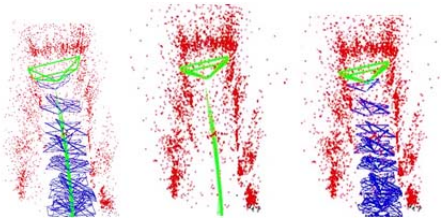

Fig. 11: Result of Real-Time Environment Mapping with essential graph (green line), keyframes (blue) and point cloud (red)

Based on the result obtained in Figure 11, the result shows that the mapping process that has been done by using the setting 2000 Features per image, 30 features point value needed to initialize and the minimum threshold value for features detection of 11 on a real-time environment. However, the 2000 Features points per image time processing may become an issue as its processing time is the longest compared to all the selected parameter settings. Therefore, 1500 Features point can be used to reduce the time processing for the mapping because the 1500 Features per image has the result almost the same as the 2000 Features per image.

*vi.* Discussion

Based on the study carried out, the most accurate parameter obtained is the 2000 Features point detection per image. It has the lowest ATE error, making it the most accurate trajectory result compared to the other parameters. However, the tracking time may become an issue for the detection because it needs a longer processing time to process all the points that are being detected [19]. The number of feature points for initialization can be lowered to make the detection much faster to overcome those issues, and coding can be adjusted to make the mapping process faster. On the other hand, the 800 Features point per image may have the lowest error among all the settings, but the produced map does not have enough information as the other settings.

During building the ORBSLAM packages, there were some problems with the ORBSLAM algorithm file and other testing algorithms. Algorithms such as LSD slam and DSO slam have been installed on the Jetson Nano(B01) for testing purposes. Still, a few issues, such as multiple dependency packages for each algorithm, make it hard to build together on the same storage. Therefore, only one algorithm is chosen as the main algorithm, the ORBSLAM2 algorithm. ORBSLAM2 is being selected as the main algorithm as it has the fastest detection and least trajectory error, according to previous research that has been done.

A few disadvantages of using the camera-based algorithms have also been found during the testing. One of the disadvantages of using any camera-based algorithm is the lighting intensity for the environment can heavily affect

the result [20]. A direct based algorithm such as LSD SLAM is heavily affected by the light intensity as it cannot process the image projection correctly. The ORBSLAM algorithm result is not affected by the result. However, if there is not enough light in the surrounding, the ORBSLAM may not be able to detect the features point of the environment as it cannot see the edge or corner of an object to do the detection. Other than that, fast rotation also happens to be a problem for the Visual SLAM algorithm as it cannot translate fast rotation that is done by the camera fast enough as the real-time[20]..

**Conclusion**

The objectives set at the beginning of the project are to analyze the performance of the selected VSLAM algorithm based on the accuracy of the map produced and to design and build suitable coding for the robot to move to map all the surrounding environments. ORBSLAM performances are being analyzed by changing the parameter used to set the detection of the environment. The most optimized performance that is being obtained is by using the 2000 Features point per image detection, which is the setting that gives the right amount of information for the environment detection and lower error compared to the other parameter settings. The most optimized setting makes the trajectory plot much more detailed, bringing a more accurate map. However, the time to track the features point at each frame also increased, resulting in much more time to complete the whole mapping process.

The robot car made has also been successfully moved by using the rosserial communication ROS coding in the Arduino nano and the Jetson Nano(B01). However, the Jetson nano (B01) cannot supply enough power to control the motor and, at the same time, execute the ORBSLAM algorithms. The Jetson nano is connected to the Jetson Nano power supply adapter to get a constant power supply to operate both processes. Despite that, both of the objectives are still being achieved, although the result for the robot movement is not perfectly made using a conventional power supply.

**Future Work**

For future works, to obtain much more precise and accurate results, a higher accuracy camera such as Intelrealsense and the Stereo camera can replace the Logitech c270 camera that is being used. It will provide a wider and more accurate view of the results. Plus, a much higher angle camera view can cover up to 180° of view of the environment [18].

The ORBSLAM algorithm code adjustment can be made by decreasing the detection, tracking, and localization time to make the mapping process faster and perform a much higher detection speed. The Deep Learning process also can be included to find the optimal setting for the mapping based on the time for tracking and the essential graph produced [21].

The robot can be improved for the robot components by adding the dc motor with an encoder that can read the odometry value of the environment more accurately. The robot can also be set up with a dependable supply that can support the jetson nano power required to operate the VSLAM algorithm. Lidar also can be used in combination with the camera to get a much more robust result [21]

***Acknowledgement***

*The author would like to thank Centre for Research and Innovation Management (CRIM), UTeM for sponsoring this work under project : INDUSTRI (MTUN)/ENDSTRUCT/ 2021/FKEKK/I00057.*

REFERENCES

[1] D. F. Wolf and G. S. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Auton. Robots*, vol. 19, no. 1, pp. 53–65, 2005, doi: 10.1007/s10514-005-0606-4.

[2] E. A. Sholarin and J. L. Awange, "Global navigation satellite system (GNSS)," *Environ. Sci. Eng. (Subseries Environ. Sci.*, no. 9783319276496, pp. 177–212, 2015, doi: 10.1007/978-3-319-27651-9_9.

[3] F. Gul, W. Rahiman, and S. S. Nazli Alhady, "A comprehensive study for robot navigation techniques," *Cogent Eng.*, vol. 6, no. 1, 2019, doi: 10.1080/23311916.2019.1632046.

[4] S. Pillai, “IN MOBILE ROBOTS Sudeep Pillai,” 2017.

[5] K. Di, W. Wan, H. Zhao, Z. Liu, R. Wang, and F. Zhang, "Progress and Applications of Visual SLAM," *Cehui Xuebao/Acta Geod. Cartogr. Sin.*, vol. 47, no. 6, pp. 770–779, 2018, doi: 10.11947/j.AGCS.2018.20170652.

[6] “LSD-slam and ORB-slam2, a literature based explanation | by Jeroen Zijlmans | Medium.” [Online]. Available: https://medium.com/@j.zijlmans/lsd-slam-vs-orb-slam2-a-literature-based-comparison-20732df431d.

[7] P. Latcham and C. N. Taylor, "Comparison of Visual Simultaneous Localization and Mapping Methods for Fixed-Wing Aircraft Using SLAMBench2," *2020 IEEE/ION Position, Locat. Navig. Symp. PLANS 2020*, pp. 1578–1586, 2020, doi: 10.1109/PLANS46316.2020.9109945.

[8] S. Valladares, M. Toscano, R. Tufiño, P. Morillo, and D. Vallejo-Huanga, "Performance Evaluation of the Nvidia Jetson Nano Through a Real-Time Machine Learning Application," no. January, 343–349, 2021, doi: 10.1007/978-3-030-68017-6_51.

[9] D. Prokhorov, D. Zhukov, O. Barinova, K. Anton, and A. Vorontsova, "Measuring robustness of Visual SLAM," *Proc. 16th Int. Conf. Mach. Vis. Appl. MVA 2019*, 2019, doi: 10.23919/MVA.2019.8758020.

[10] R. Vilches, I. Martinez, M. A. L. Gonzalez, J. Crespo, and R. Barber, "Mobile Robotics Teaching Using Arduino and Ros," *Iceri2014 7th Int. Conf. Educ. Res. Innov.*, no. November, pp. 827–833, 2014.

[11] A. Marin-Hernandez, "ROS Tutorial : Robotics Operation System," *LAAS-CNRS Robot. Action Percept. Gr.*, pp. 5–7, 2014.

[12] A. Chatterjee, N. N. Singh, and A. Rakshit, *Vision Based Autonomous Robot Navigation ; Algorithms and Implementations*, vol. 2, no. 10308. 2013.

[13] N. Sünderhauf, "Robust Optimization for Simultaneous Localization and Mapping," *Disseration Chemnitz Univ. Technol.*, no. April 1981, pp. 1–231, 2012.

[14] J. Kacprzyk, *Robot Operating System (ROS) Vol 5*, vol. 5, no. Volume 5. 2021.

[15] M. Haddad, S. Cheng, L. Thao, and J. Santos, "Autonomous Navigation Powered by Jetson TX2 and Robot Operating System," pp. 1–30, 2019, [Online]. Available: https://canadacollege.edu/stemcenter/documents/aspiresinternship2019/ASPIRES S19 Final Paper EE.pdf.

[16] M. Filipenko and I. Afanasyev, "Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment," *9th Int. Conf. Intell. Syst. 2018 Theory, Res. Innov. Appl. IS 2018 - Proc.*, no. September, pp. 400–407, 2018, doi: 10.1109/IS.2018.8710464.

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

[18] I. Z. Ibragimov and I. M. Afanasyev, "Comparison of ROS-based visual SLAM methods in homogeneous indoor environment," *2017 14th Work. Positioning, Navig. Commun. WPNC 2017*, vol. 2018-Janua, no. July 2018, pp. 1–6, 2018, doi: 10.1109/WPNC.2017.8250081.

[19] D. Scaramuzza and F. Fraundorfer, "Tutorial: Visual odometry," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, 2011, doi: 10.1109/MRA.2011.943233.

[20] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: A survey from 2010 to 2016," *IPSJ Trans. Comput. Vis. Appl.*, vol. 9, 2017, doi: 10.1186/s41074-017-0027-2.

[21] B. Huang, J. Zhao, and J. Liu, "A Survey of Simultaneous Localization and Mapping with an Envision in 6G Wireless Networks," no. October, 2019, [Online]. Available: http://arxiv.org/abs/1909.05214.