SIMULTANEOUS LOCALISATION AND MAPPING (SLAM) TECHNIQUE IN REAL TIME: AN INTRODUCTION OF DIK-SLAM

Olusanya Y. Agunbiade,* Mziwoxolo Mayedwa,* and Awosejo Oluwaseun Johnson**

Abstract

The issue of simultaneous localization and mapping (SLAM) has been thoroughly investigated in robotics. Its influence on independent robot navigation attracted scholars. Over decades, a variety of approaches have been suggested to handle the SLAM problem with commendable success, however there are a variety of factors that can reduce the efficiency of the SLAM techniques. Environmental elements, such as lighting conditions, shadow, dynamic (non-static) conditions, kidnapping event, computational complexity, and shadows are some of these concerns (factors). These challenges (factors) produce inconsistencies, which might result in execution that yields undesirable results. In an attempt to overcome these challenges, a unique SLAM approach identified as DIK-SLAM has been presented. The dynamic illumination and kidnapping (DIK) SLAM technique is an improved version of the Monte-Carlo (MCL) SLAM algorithm that incorporates filtering techniques and various adjustments to increase the reliability and considering computational cost. The normalised differences index (NDI) is the filtering approach used by the DIK-SLAM to eliminate shadow. To minimise the effect of light intensity, filters like specular-to-diffuse and dark channel models were also applied to the DIK-SLAM. Given that the computational cost is a consideration, these filtering techniques are running concurrently. In addressing the kidnapping problem and the dynamic (non-static) environment, respectively, the revised MCL algorithm founded on grid map and initial localisation approach was presented to create the DIK-SLAM. In this article, the SLAM algorithms were evaluated using a publicly released dataset (TUM-RGBD). The MATLAB simulation software was used to conduct the test, and results were evaluated quantitatively. Thus, comparing the DIK-SLAM, the traditional MCL algorithm, and other SLAM approaches accessible in the literature, experimental results showed that the DIK-SLAM performed better because for most of the trajectory evaluation it attained lower error. The DIK-SLAM technique presented in this paper has the ability to support independent movement, route planning, and exploration while minimising the robot failure rate, injuries and accidents to humans.

Key Words

Independent vehicle, dynamic condition, lighting condition, kidnapping event and shadow

1. Introduction

Autonomous navigation, which can be accomplished by using simultaneous localization and mapping (SLAM), plays an essential role in assisting movement of independent vehicle from one position to another without human intervention [1]. These properties have caught the attention of scholars, and have been successfully improved from an organised landscape towards many unsafe conditions, including unorganised landscape and underwtaer landscape [2]. Considering a defined environment, specific information might very well support in appropriate preparations for future movement/motion. However, in such an unfamiliar environment, the current pose measurement is necessary for generating the environment map [3]–[5]. This generated map would then be used to facilitate navigation and this signifies the core principle of SLAM [3]. The issue is determining how to apply SLAM to measure pose of the robot and develop the map for an environment. In the scope of SLAM, data gathered from sensors (camera, laser, sonic, and radar) is applied to evaluate the robot pose and the estimation of map. This method is generally productive because the robot's pose estimation is directly linked with the feature estimation in the environment [3]. In past studies, several strategies to mapping have already been proposed, but they can be broadly classified as feature-based or grid-based techniques [3]. These strategies had also been shown to be effective, but their choice is heavily dependent on the type of the environment and both have limitations. The feature-based method

* University of the Western-Cape, Bellville, South Africa; e-mail: agunbiadeolusanya@yahoo.com
** Tshwane University of Technology, Pretoria, South Africa
Corresponding author: Agunbiade Olusanya Yinka

Recommended by Gian Luca Foresti
(DOI: 10.2316/J.2024.206-1045)

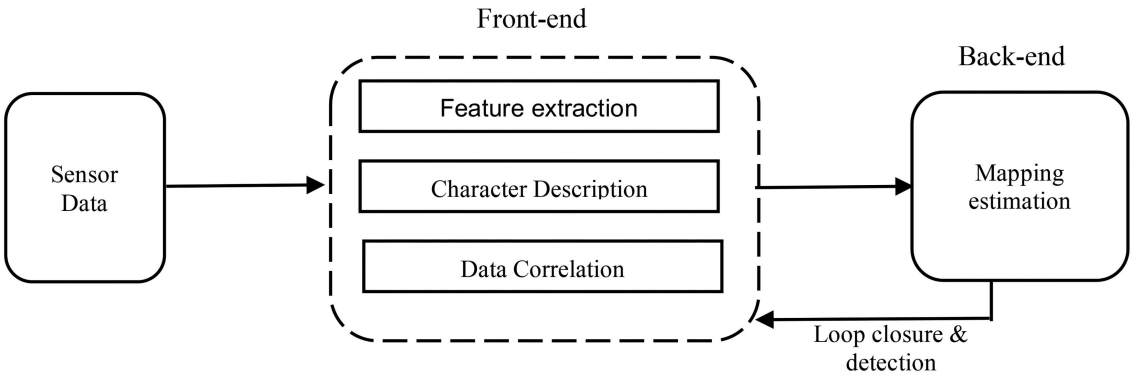

Figure 1. Overview of the visual SLAM [6].

is commonly recommended in large open surroundings with a predetermined object and without these objects, the SLAM performance depreciate [3]. The grid-based technique works better in densely clustered environments amd can handle arbitrary objects but suffers from exessive computational complexity and massive memory usage [3]. Sadly, there are also other issues and to highlight them, the study undertook a broad evaluation of several SLAM methodologies put out by earlier researchers. This review discusses different SLAM algorithms, their strategy, drawbacks, and benefits. According to the review, the dynamic environment, varying lighting, loop closure, computational cost, and kidnapping robot are ongoing issues that prevent society from fully embracing SLAM. Thus, solutions to these issues were thought about, and a dynamic illumination and kidnapping (DIK)-SLAM that took into consideration the computational cost was developed. The DIK-SLAM is a Monte-Carlo (MCL) algorithm that has been modified and upgraded with numerous algorithms. Successful deployment has helped to further SLAM research, and industry acceptance will increase output and safety for human working in dangerous environment. The structure of this article is as follows: The literature review is discussed in Section 2, the framework is discussed in Section 3, while Section 4 present the experiments and the results, and the last Section present the conclusion and future work.

2. Literature Review

Autonomous navigation let robots navigate successfully on their own with no human assistance, which is a crucial feature for independent expeditions into unexplored environment. Several scholars have presented numerous SLAM approaches in recent years with excellent results [4]–[7]. To identify current issues, reviews were conducted on various SLAM techniques to discuss their methodology, experiment, result, and constraints.

The study of [6] discussed a visual SLAM technique that rely on data collection, front-end operation, back-end operation, and loop detection to implement SLAM. The front-end operation assists with estimating the camera's relative motion using information from adjacent images to generate local images. The objective of the back-end operation is to improve the initialisation information received from the front-end operation using statistical inference. Loop detection is used to determine if the preceding scene has been visited again through visual detection of posture information at a specific point in time. The visual SLAM technique framework is presented in Fig. 3.

The experiment shows that the visual SLAM techniques performs very well, it was able to cope with loop closure and kidnapping, but the use of the assumption on invariance of pixel grey scale actually makes it susceptible to environmental lighting, camera exposure, and other factors, limited the performance of the system.

In the study of [7], they discussed a camera-based semantic SLAM technique for low-cost cars. In their research, a local semantic map was generated by combining the CNN-based semantic segmentation results and the optimised trajectory after pose graph optimisation. A compacted global map was then generated (or updated) in the cloud server for further end-user localisation based on the ICP method and within an EKF framework. The average size of the semantic map was 36 kb/km. Experiment evaluation of the camera-based localisation framework proof to be reliable and applicable to autonomous driving. However, the technique when confronted with moving objects within the road environment caused a drift of perception, localisation, and mapping for autonomous robot. Thus, future work plans to address the problem of detection and tracking of moving objects (DATMO) by introducing algorithms that can detect and handle features that are not stationary in the environment.

The study of [8] discuss a Kalman filter SLAM technique that function based on four steps. The first step represents the observation stage where the robot extracts various features *e.g.* doors, line *etc.* from sensor data. The second step represents the measurement stage where the robot will generate a measurement for its feature's observation from its estimated positions that are from the outcome of the prediction step. The third step represents the matching phase, where robot estimate the best match between features extracted from its observation measurement and the features selected during the measurement prediction. The fourth step represents the estimation phase where the robot updates it belief state by fusing the matching information. The experimental

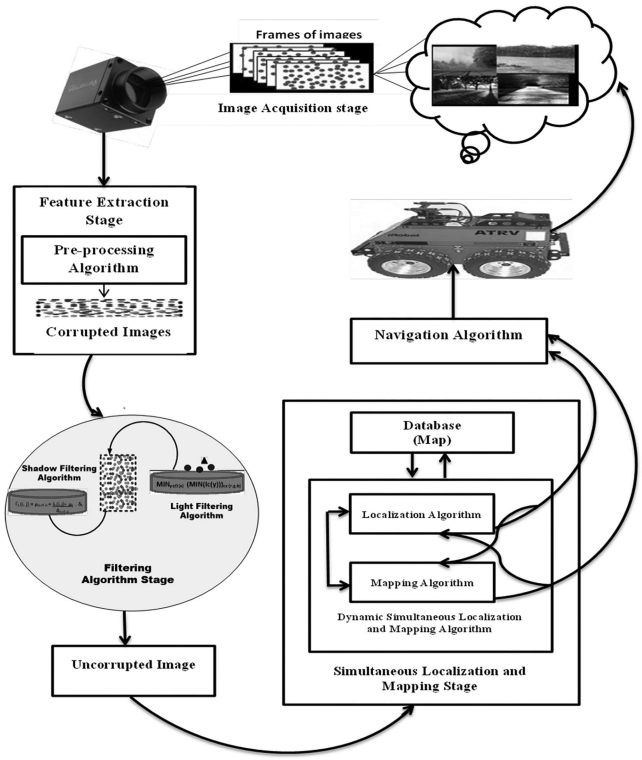

Figure 2. The DIK-SLAM algorithm model.

performance of the system is impressive. However, as stated in their research, the robot initial position with a certain approximation must be known in their Kalman filter approach at all time. Therefore, if the robot gets lost, it can't recover its position. Thus, their SLAM technique is limited to function in global localisation and kidnap robot problem. The review given in this paper led to the conclusion that investigations should focus on challenges, such as illumination variation (light intensity and shadow), kidnapping robots, computing expense, and non-static/dynamic objects in the environment. This research, however, developed a DIK-SLAM technique taking into account computational cost with several modifications to solve each of these drawbacks. Filters will be introduced into DIKSLAM to address the environmental noise, occupancy grid technique was introduced used to overcome dynamic environment, while initialise localisation and appearance matching was introduce in to DIKSLAM to address kidnapping and looping.

3. Methodology

The proposed SLAM technique has five phases which are the image acquisition phase, feature extraction phase, filtering level, SLAM phase, and the navigation phase [9]–[26] and they will be discussed in this section. Thus, Fig. 2 depicts the workflows for all phases of the DIK-SLAM, which has been developed in MATLAB and was derived from the study presented in [35]. MATLAB is a 4th programming language that can perform mathematical computation and was widely utilised due to its image processing capabilities [34].

3.1 The Acquisition Process

The image acquisition/captured phase within the domain of imaging and computer vision converts received visual data into an electrical signal which could be interpreted and read [31]. Such data is known as stream digital images and could be generated by a wide range of devices, such as cameras, radar sensors, and laser scanning devices, among many others. Consequently, the DIK SLAM incorporates a camera device to assist with the way of translating real-world input into a readable and understanding entity. The camera is considered for such a phase because it can collect much surrounding data than any other sensor, which helps to improve the robustness of the SLAM technique in decision-making [31]. Although it helps with data collection, this step also serves as a major reason why environmental disturbances (Shadow and light) and dynamic features appear in the image.

3.2 Features Extraction Phase

Stage two of the DIK-SLAM technique is the feature extraction phase. This is a procedure that employs statistical techniques and several filters to collect characteristics from several image sections (drivable section, non-drivable section, and uncertainty) [29]. Image features for SLAM techniques include color, texture, and border, among others. In MCL SLAM, unless the robot extracts data from its surroundings, it is difficult for the vehicle to generate the precise position and orientation for itself using the provided model/map [10]. The collected data from the environment is known as belief, as represented in (1) [10].

$$\mathrm{bel}\,(s_t) = p\,(s_t \backslash z_t, u_t) \tag{1}$$

where z_t signifies the sensor measurement, u_t signifies the control state, and s_t signifies the state sample at a time t [31]. The belief distributions, which are iteratively constructed using control and measurement data, provide a useful statistical technique for addressing the SLAM problem [10].

In the MCL technique, successive belief is defined by a collection of samples that are utilised to identify state features that might help direct robot direction, this sample is represented in (2) and they are hypotheses for object identification, such as obstacle, borders, and walls are orthogonal, as presented by [10], [11].

$$S_t = \left\{ s_t^{[n]}, w_t^{[n]} \right\}_{n=1,\ldots,N} \tag{2}$$

However, a few samples violate the object representation hypothesis when faced with environmental noise, making it difficult to identify the object in the image and perhaps leading to SLAM failure [29].

3.3 Filtering Algorithm Phase

The first alteration to the revised MCL is to improve its capability to withstand some external factors that might impact image characteristics, which in turn could result in inaccurate pose estimation or, in extreme cases,

result in kidnapping with no chance of recovery in the scenario of increase in measurement noise [29], [35]. These external factors (light and shadow) do have the potential to deteriorate the image, cause vision problems, and disrupt the colour element valve, resulting in a negative impact on the object/feature extraction [29]–[35]. The third step of the SLAM approach, the filtering stage, was incorporated into the system to reduce the impact of light intensity and shadow on the feature extraction stage and improve classification performance at the SLAM phase. Due to the frequent occurrence of environmental noises like light intensity and shadow, two filters were developed to identify and eliminate them. In Sections 3.3.1 and 3.3.2, these filtering algorithms are presented.

3.3.1 Shadow Filtering Algorithm

The shadow impact in an image can be minimised by using the Shadow filtering algorithm. The filtering functionality is dependent on morphological operations and the normalised differences index (NDI). A shadow maintains easy identification with the maximum value of saturation (S) and the minimum value (V) in HSV colour space [35], therefore, the image RGB valve is primarily transformed to HSV using (3)–(5) [33], [35].

$$V = \frac{1}{3}(R+G+B) \tag{3}$$

$$S = 1 - \frac{3}{(R+G+B)}\min(R,G,B) \tag{4}$$

$$H = \begin{cases} \theta & \text{if B} \leq \text{G} \\ 360^o & \text{if B} > \text{G} \end{cases} \tag{5}$$

where

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\},$$

The colours red, green, and blue indicate RGB. The shadow noise region is extracted using the saturation (S) and value (V) elements of the I image [32], [35]. Using the NDI, (6) provides an illustration (NDI) [33].

$$NDI = \frac{S-V}{S+V} \tag{6}$$

OTSU threshold (T) algorithm is applied to segment the NDI images to ensure that regions classified as shadow regions are truly shadow regions [31]. The next step is to estimate the buffer area ($I_{\text{buff},\,k}$) which is defined as a non-shadow pixel around the shadow region to relight the pixel of the shadow area to obtain a shadow free image [12]–[35]. The shadow removal concept relies on the transformation function signifies as $I_k^i(i,j)$ in (7). This depicts the mean and variance of the buffer region around the shadow that was employed to correct the shadow pixel [29]–[35].

$$I_k^i(i,j) = \mu_{\text{buff},k} + \frac{I_k(i,j)-\mu_K}{\sigma_{\text{buff},k}} \tag{7}$$

where $\sigma_{\text{buff},\,K}$ and $\mu_{\text{buff},\,K}$ represent, respectively, the variance and mean of the image I at a buffer position ($I_{\text{buff},\,K}$). σ_K and μ_K represent, respectively, the variance and mean of the shadow pixels image I in the position I_k [32]–[35].

3.3.2 Light Filtering Algorithm

The light filtering algorithms may handle the influence of light intensity created by sunshine, which is a frequent noise given the excess brightness of sunlight. This removal approach for light effect is evaluated by modelling the object reflected by a colour camera using the dichromatic reflection technique, which comprises the diffuse and specular reflection elements given as $I(x)$ in (8) [31].

$$I(x) = I^D(x) + I^s(x) = w_d(x)\,B(x) + w_s(x)\,G \tag{8}$$

I depicts the intensity of the observed image and the image coordinate is $x = \{x,\ y\}$ [35], and diffuse reflection element symbolises I^D while the specular reflection element symbolises I^S, G and $B(x)$ represent the specular colour and the diffuse colour, respectively. $w_d(x)$ and $w_s(x)$ denote the coefficients that determine the magnitude of the diffuse and specular reflection elements [35].

The presented light intensity detection is relying on the dark channel $\left(I^{\text{dark}}(x)\right)$ defined in (9) and optimal automated thresholding is use to determine the high light region in an input image [35]. This approach is combined to properly classify the area of the image influenced by light intensity [35]. According to the dark channel model, regions that are impacted by strong light will exhibit a high peak value, while regions that aren't impacted would have a low-intensity value [26].

$$I^{\text{dark}}(x) = \min_{y\in\nu(x)}\left(\min_{c\in(r,g,b)}(I^c(y))\right) \tag{9}$$

The $\nu(x)$ signifies the local patch positioned at x and the image coordinate is denoted as image coordinate while the I^c is represented as the colour channel [35].

The OSTU thresholding is the algorithm used to label the high light (peak) area in the image. The labelled image [mark(x)] created by the automatic thresholding of the dark channel image is labelled with a 1 to indicate the region that is influenced by light intensity as well as 0 to indicate the unaffected regions with light intensity [35]. Equation (10) serves as an illustration for the expression [32]. The t^* signifies the ideal threshold for classifying a mark image (x).

$$\text{mark}(x) = \begin{cases} 1 & \text{if } I^{\text{dark}}(x) \rangle\ t^* \\ 0 & \text{Otherwise} \end{cases} \tag{10}$$

The elimination of the intensity of light is based on specular-to-diffuse theory [26]. Equation (11) illustrates an image with no impact of light intensity [33]–[35].

$$I^D(\wedge_{\max}) = I - \frac{\max_{u\in(r,g,b)} I_u - \wedge_{\max}\sum_{u\in(r,g,b)} I_u}{1-3\wedge_{\max}} \tag{11}$$

However, incorporating such filter techniques results in significant computation costs [33], but these constraints would be handled by the rotating tilling (RT) methodology, which is detailed in Section 3.3.3. Given that the computational cost is taken into consideration, this approach is utilised to speed up the execution of the filtering process [13]–[29].

3.3.3 Concurrent Approach

A recommended mechanism for filters will enables multiple processes in a short period of time [21]. Concurrency originally was designed to solve the limitations of the conventional technique (serial operation) with slow processing capabilities. There are other concurrency approaches, however, the two most popular are binary swapping (BS) and parallel pipelined (PP) [13]–[35]. The ring rotation topology is used in parallel pipelining to accomplish the partial composition. This enables the usage of any number of processors, however with more extensive communication procedures. Thus, BS is only possible whenever the processor's numbers are restricted to a power of two, with lower communication steps than that of the parallel pipeline approach [13]–[35]. We used RT in this study with two processor because we dealing with two filtering algorithms. The RT leverages the strengths of both BS and PP methods to address their constraints. This was accomplished by the fact that the processor configuration uses a ring rotation topology while the data communication method is dependent on the BS technique. In the RT approach, the image is processed in three steps, which are covered in Sections 3.3.3.1 through to 3.3.3.3.

3.3.3.1 Image (Data) Partitioning Stage

Distributing image volume among processors and reducing computing costs are the goals at this stage [13]. There are data partitioning step of the RT approach, any effective data partitioning approach may be used, and there are several variations of this approach [13]. In [14] uniformly allocates image segments among processors, however, this causes significant overhead communication and excess time complexity during the image composition step. While volume data partitioning presented by [14]–[35], has comparable features regarding allocation among processors with little overhead communication and extra computational costs during the image composition stage. Although the sharing volume data partitioning described in [14] reduces overhead communication with reduced computing time during the image composition stages, image dimensions weren't divided equally among processors. Since computation cost is taken into consideration, the sharing volume data partitioning described in [14] was used because it supports reduced overhead communication and computing time.

3.3.3.2 The Rendering Procedure

This represents the second step following the procedure of splitting the dataset (image), which produced a partial image according to the number (2) of processors [14]–[35]. The rendering procedure is applied to every partial image allocated to each processor. The rendering technique encodes, resamples, and generates the initial block that corresponds to the partial image [13]. In this process, initial blocks are formed individually by each processor, and overlapping is avoided in the procedure by numbering block images pertaining to each processor prior to the dispersion among associated processors (2) [13]. The dispersion makes it possible to identify and eliminate unwanted environmental noises at the same time. In the RT procedure, the processor (P_r) sends the block size $\left(A_r^k(m)\right)$ of its own partial image to other processors (P_i) based on (12) and accepts another block size $\left(A_j^k(m)\right)$ of another partial image from another processor (P_j) based on (13) [13], [14]–[35].

$$P_r\left(A_r^k(m) \rightarrow P_1\right) \text{ where}$$
$$\begin{cases} l = 0, 1 \ldots, k-1 \\ w = 0, 1 \ldots, \lceil P/N \rceil - 1 \\ v = 0, 1, \ldots, \lceil P/2^k \rceil \\ m = \left(\left(r - 2^{k-1} + 2l\right) \bmod 2^k\right) 2^{k-1} + 2^{2k-1}v + P \\ i = (r - 2^{k-1} + l) \bmod P \end{cases} \tag{12}$$
$$P_r \leftarrow P_j\left(A_j^k(n)\right), \text{where}$$
$$\begin{cases} l = 0, 1 \ldots, k-1 \\ w = 0, 1 \ldots, \lceil N/P \rceil - 1 \\ v = 0, 1, \ldots, \lceil P/2^k \rceil \\ m = \left((r + l) \bmod 2^k\right) 2^{k-1} + 2^{2k-1}v + Pw + 2l \\ i = (r - 2^{k-1} - l) \bmod P \end{cases} \tag{13}$$

The j, r and i signifies the ranks of the processor, k represents a positive integer while the m and n signifies the blocks numbers [13]–[35].

3.3.3.3 The Composition Procedure

This represents the final stage, following the rendering procedure that produced some number (4) of initial blocks [35]. However, at this phase, every processor uses the Over-operation to combine the block of image it got during distribution. MPICH, a collect command of a message-passing library on multicomputer memory allocation, is employed to merge every block till the final outcome (filtered image) is formed [14]–[35]. This complete image is generated with minimal impact of the environment noise. The RT procedure has a number of communication step and for this study 16 communication step was realised but the major advantage is that, at each communication stage, all algorithms are detecting and eliminating environmental noises simultaneously as result limited processing time is required during the re-generation of the complete image. Thus, detail information about the composition procedure is provided in one of our study [35].

3.4 Simultaneous Localisation and Mapping

Localisation indicates a vehicle's capacity to recognise its very own direction and position inside an environment,

and mapping indicates a vehicle's competency to design a representation model of an unfamiliar territory [12]–[15]. These two procedures are crucial for the robot's navigation. Localisation and mapping are two independent operations in robotics, though they are closely linked since the intelligent vehicle requires the map to locate itself, as well as the vehicle's precise location is required map creation [15]. Hence, the SLAM system has captured the interest of several academicians, due to its possibility of tackling the relationship between the two problems (Mapping and Localisation) which can facilitate independent navigation. The SLAM algorithm has been improved over time with good performance while being predominantly used in static environments [16], [17]. Nevertheless, in this work, we improved the SLAM approach to handle either static or dynamic (non-static) environments. Additionally, as shown in Fig. 2, the database was used to execute the initialising localisation, similarities, and dissimilarities comparisons. This really is essential in handling both loop closure and kidnapped robots. The traditional MCL approach was modified to address the SLAM problem at this phase, as discussed in Sections 3.4.1–3.4.3.

3.4.1 The 2nd Revision to Present the DIK-SLAM Algorithm

The DIK-SLAM framework uses the basic MCL technique for pose estimation (localisation) and map construction, but it has to be updated to handle dynamic environments, kidnap robots, and looping closure. The MCL procedure is a probabilistic approach that is widely employed to handle the challenge of SLAM. Unlike some of the other high-computational cost algorithms, such as EKF and Kalman filters [18], this constraint has diverted investigators' focus to particle-based approaches, that have produced satisfying results [29]. In this work, we proposed using the MCL algorithm, a particle-based technique limited to a static environment [19]. Thus, coping in non-static/dynamic situations needs the MCL method to be updated. The major purpose of this subsection is to go through the DIK-SLAM update which enables the technique to cope with changing (dynamic) environments because of the presence of moving objects and handling such issues requires modifying the map as the environment state changes. The updated MCL algorithm is built on the idea that every cell of a map is treated as a separate object, as demonstrated in (14).

$$Y_t = (\text{cell}_1, \ldots, \text{cell}_{n,t}) \tag{14}$$

A collection of separate cells is referred to as y_t. Given that the cells are independent, the observation for such new status $p\,(y_t, x_t\,|z_t, u_t\ )$ is being modified simply by adding probabilities of the cells map to the original MCL whenever a cell's state changes. This factorisation approach is comparable to the one described in article [35] for adding a door's status into MCL technique. Thus, the factorisation for this stage in full details is presented in one of our previous research and it is publicly accessible [35]. Thereafter, the factorisation of this phase, an incorporates of a new probability of the cell is added to traditional MCL, whereas the cell's state is unknown during this stage. Reference [35] presents a more extensive method of integrating the new probability of the cells into the traditional MCL. However, estimating the updated probability of the cells assists in determining the state of the cell introduced to MCL, which is important in identifying dynamic situations, as discussed in Section 3.4.1.1.

3.4.2 1 The Binary Object Representation Using Bayes Algorithm

At this point, MCL, a recursion Bayes filter, was already improved to deal with a new likelihood for every cell on the map, however, computing the state for such a cell hasn't yet been finalised. This subsection will go through how to estimate the probability of every cell inside the map, which is necessary for defining the cell's state and is crucial in working in a dynamic environment [35]. The cell probability estimate technique used in this study is comparable to the study presented by [35]. The approach represents cells as binary variables that could be unoccupied in the absence of an object and occupied in the present of an object. Furthermore, a cell's state is either 0 or 1, with 0 indicating that the cell is empty (not occupied), and 1 indicating the opposite when the cell is occupied. This improvement has enabled DIK-SLAM to cope with dynamic environments by employing the grid-Map approach, which focuses on cell state during localisation. The detail factorisation and full description on how the original MCL is enhance to handle the status of the cells is presented in one of our previous research and is publicly available [35].

3.4.3 The 3rd Revision to Enhance DIK-SLAM Ability to Detect and Recover From Kidnapping

The provided DIK-SLAM method was improved once more to tackle the problem of a kidnapped robot. The kidnapping dilemma arises in SLAM whenever the robot takes an unforeseen movement in its environment while being unaware of it. This difficulty arises whenever a sensor stops working properly or when measurement noise becomes severely high [20]. As a result, a robot could struggle to measure its location, this violates the hypothesis of resolving the SLAM issue as pose estimating is essential for map formation and therefore may give rise to failing without recovery [31]. The approach which might handle such a challenge must be capable of achieving these three aims: pose estimate, detection of kidnapping, and global localisation [20]. Mostly in research, scan to map match is a frequent strategy for handling kidnapped robot. This approach as presented in [13]–[35] uses a matching algorithm to perform an in-depth check of a present observation across the reference map (pre-define mapped environment) to establish the present pose estimation with respect to the specified referenced map. This strategy, unfortunately, might not work throughout all instances. For example, selecting a good match from a reference map could be difficult if the present observation has transformed because of dynamic features. Thus, matching will become

unachievable and can result in kidnaping with no chance of recovery [21]. This is a real-world possibility, which was considered in the DIK-SLAM architecture. In the DIK-SLAM, the current observation image and the reference image will be compared and contrasted at the earlier phase before beginning the re-localisation process. Whenever robot kidnapping occurs in a particle-based technique like MCL, the particle drifts from local samples to global samples, and thereafter re-localisation, the global samples will then move back to the local samples [22]–[35]. The global samples are critical for resolving the issue of recovery and kidnapping in robot [22]. More so, the probability of these particles was also taken into account. If the maximal probability for the particle has a lower value than the coefficient, the vehicle initiates a kidnapping. This method was utilised in paper [22] but was applied in DIK-SLAM, however, it centred around particle weight. The formulation for kidnapping in DIK-SLAM is shown in (15).

$$\text{Kidnapped robot}_{\,t} = \begin{cases} 1 & w_t^{\max} < \gamma \\ 0 & \text{Otherwise} \end{cases} \tag{15}$$

Where $w_t^{\max}$ symbolises the particle's maximal probability.

In an event of kidnapping, the scanning to matching strategy employs the SIFT descriptor to evaluate the present (current) and reference images to that of an earlier map that's going to be utilised during re-localisation. The approach was recommended by [13], and it was applied in this study because of its successful outcomes. Furthermore, the Fourier signatures of the similarities and dissimilarities measures were added to the DIK-SLAM once more to account for image similarities measurement, as shown in (16) and (17) [23].

$$\text{Disim}\ (I_i,\ I_t) = \sum_{y=0}^{l-1} \sum_{k=0}^{m-1} |F_{\text{i y}}(k) - F_{\text{t y}}(k)| \tag{16}$$

$$\text{Sim}\,(I_i,\ I_t) = 1000 - 1000 \frac{\text{Disim}\,(I_i,\ I_t) - \text{Min}_{\text{i t}}\{\text{Disim}\,(I_i,\ I_t)\}}{\text{Max}_{\text{i t}}\{\text{Disim}\,(I_i,\ I_t)\} - \text{Min}_{\text{i t}}\{\text{Disim}\,(I_i,\ I_t)\}} \tag{17}$$

I_i symbolises the reference image and I_t denotes the present (current) observation image, while $F_{\text{i y}}(k)$ symbolises the Fourier coefficients of the k^{th} frequency of y th row image in the reference image and $F_{\text{t y}}(k)$ denotes the Fourier coefficients of the k^{th} frequency of y th row image in the present (current) observation images [35].

The higher the Fourier similarity value, the more similar the reference and actual (Current) observation images [23]–[35]. Therefore, based on the matched map for both reference and actual observation (current) images, the robot's pose is being measured to address the SLAM problem. Furthermore, as the Fourier dissimilarity value increases, both reference, and actual (current) observation images become increasingly dissimilar [23]. In this case, the selected matching map isn't going to be applied. Alternatively, we offer another strategy that entails generating a new sample collection from the actual (current) observation image for the re-localisation process since the disparity between both the reference and actual observation images may not give a satisfying result [35]. The samples collected may be utilised to create a map that will assist in evaluating an accurate vehicle pose which would proceed to assist the vehicle in navigating till it emerges in a predetermined environment (no kidnapping), during which now the vehicle can use its reference map to proceed with pose estimation. Thus, this modification has a constraint that raises the computing complexity.

3.4.4 The 4th Revision to Enhance the DIK-SLAM Ability to Detect and Close Looping

The DIK-SLAM was improved again for the fourth time to increase the MCL computation capacity for resolving the issue loop closure. In SLAM, close looping is a function for detecting if an independent vehicle is conscious of arriving at an earlier visited surrounding during exploration [27]. This loop closure function is comparable to kidnapping, but the robot is conscious of the environment but must know whenever it is returning to an area previously visited, and several hypotheses to detect loop closure exists in the literature [27]–[35]. The appearance-based approach, which was utilised to tackle kidnapping, was indeed employed to deal with loop closure. The concept is once the loop detection is initiated, it implies that the vehicle has returned to its prior environment [35]. Hence, the current image using the SIFT descriptor will match a corresponding reference image (created during prior navigation) inside the database and the loop has to be closed by utilising the map created from the previously matched image to extract the vehicle pose to continue the process of SLAM [27]. Provided that the vehicle has returned to a previous location, the entire posterior needs to be calculated and the factorisation in details is presented in one of our previous study that is publicly accessible [35].

3.5 Navigation Algorithm

The unmanned vehicle is relying solely on pose estimation and the map created for is environment for navigation and this was addressed by the SLAM stage. Thus, the vehicle's path, which considers the map produced as a safe path for optimum navigation from the commencement to completion, is still not addressed [5]–[24]. The situation is made worse because the robot will not have a comprehensive map of its actual environments until it receives updates from subsequent stages [25]. Consequently, anytime additional map information is acquired, the SLAM technique must review and re-plan its optimal trajectory route. There have been various versions of navigation algorithms documented in past research, which include the famous D^* and A^* [25]. The D^* approach, on the other hand, has been selected in this research for its effective heuristic use and ability to deal with constantly new map information. This D^* navigation algorithm operates on a grid-based method, dividing the map (observation region) into $m \times m$ grid. Additionally, the D^* approach employs the cost function for trajectory planning by moving through the cell with the lowest cost

Table 1
The Datasets (TUM RGB-D) Description

Environmental sequences	Brief description
Freiburg2_desk_with_person	The environment suffers from the dynamic and environmental noises
Freiburg3_no_structure_no_texture_far	The environment suffers from low object in the environment
Freiburg2_360_kidnap	The environment suffers from kidnaping and looping
Freiburg3_Siitng_static	The environment with static and slow moving object
Freiburg3_long_office_household	The environment is a large office room with visible structure and texture

function after computing the cost function for each cell [29]. Let X denotes the observation region which is split into $m \times m$ grids and for this study 2x2 was employed, the cost function $f(R, X)$ for the independent vehicle path from the present location (R) to the final location (G) is evaluated based on (18) [25]–[35].

$$f(R, X) = g(X) + h(R, X) \tag{18}$$

$g(X)$ indicates the least cost function from X to G and $h(R, X)$ symbolises the evaluated cost function from R to X [35].

4. Experiments and Results

This section outlines the assessment approaches used to evaluate the recommended SLAM technique's effectiveness. These assessment techniques are common measuring strategies used by previous scholars because of their consistency. These assessment methods were utilised to compare the findings and reveal the SLAM method with the best performance.

4.1 Experiment 1: The TUM RGB-D (Public) Datasets

This subsection assessed the proposed DIK-SLAM approach based on the dataset known as TUM RGBD, which was made accessible to the public for use. The TUM computer vision group originally owned the TUM RGBD. This dataset is generated from a Microsoft Kinect RGB-D camera sensor with the ability to estimate ground truth measurement during capturing and motion [35]. The colour camera generates a series of 640*480 images at 30 Hz video frames/s. The TUM RGB-D is a large series of RGB-D datasets with ground truth trajectory estimates for several RGB-D data sequences [35]. There are several environmental sequences in this dataset, however, only five were used for this research and are explained in Table 1.

These environmental datasets were utilised to analyse and evaluate the reliability of different SLAM methods and are obtained at `https://vision.in.tum.de/data/datasets/rgbd-dataset` [35]. The public dataset is a widely known dataset, particularly in SLAM that function on vision sensors, meanwhile this research is presenting a SLAM technique that relies on camera sensor, therefore, this dataset is adequate for testing and was employed for the experiment. In this section, the MATLAB simulation was presented for an assessment among DIK-SLAM, RGB-D SLAM, RTABMAP, and the MCL algorithm. The RGB-D SLAM and RTABMAP algorithm result for these datasets is publicly available and can be found in [28]. Likewise, the original MCL algorithm is publicly available and can be found on this site: `https://www.mathworks.com/help/nav/ug/monte-carlo-localization-algorithm.html` and in the study presented by [29]. The SLAM qualitative trajectory was tested by using the overall time indices of the translational component as linked to the absolute trajectory error (ATE in RMSE) [35]. Considering the frame (sequence) of images in the datasets, the trans (F_k) represents the translation component of the ATE for every time step (k) and m is the total number of time steps in the sequence [35]. The mathematical equation for the trajectory error summation is presented in (19) and the outcome of the error assessment in RMSE is shown in Fig. 3.

$$\mathrm{RMSE}(F_{1:\mathrm{m}}) := \left(\frac{1}{m} \sum_{k=1}^{m} \left\| \mathrm{trans}(F_k) \right\|^2 \right)^{\frac{1}{2}} \tag{19a}$$

where,

$$F_k = SP_k \left(Q_k^{-1} \right) \tag{19b}$$

F_k denotes an ATE for a particular time step (k), determined by measuring the difference between the real-life measurement that is error free and the simulated trajectories measurement.

Given the comparison experiment presented between the algorithms in Fig. 3. The MCL is characterised by a blue bar while DIK-SLAM is represented by a red bar, the RGBD-SLAM is represented by a green bar, and the RTABMAP is represented by a yellow bar. In the Freiburg2_desk_with_person which represent the environment that suffers from a dynamic environment due to the presence of moving object and environmental noise due to the present of illumination variation. Thus, the DIK-SLAM had the best performance with lesser trajectory ATE (RMSE) while the RGBD-SLAM is the second-best performance followed by RTABMAP and the traditional MCL algorithm. This implies that the DIKSLAM is able to handle dynamic condition and environmental noise better than other algorithms and this is because of the

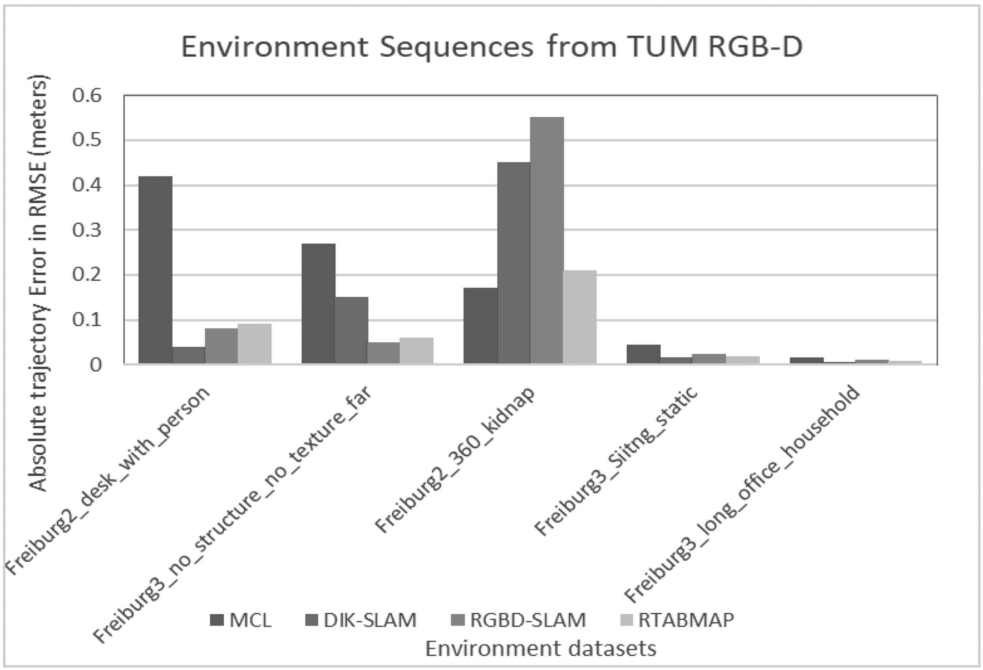

Figure 3. The absolute trajectory error (RMSE) in meters.

filters and the map-grid technique present in the DIK-SLAM. The traditional MCL algorithm had the worst performance because it was created to function in a static environment which also explains why the RMSE of ATE is very high compared to other algorithms. In the Freiburg3_no_structure_no_texture_far which represents the environment with a low object. Thus, the RGBD-SLAM had the best performance with lesser trajectory ATE (RMSE) while the RTABMAP is the second-best performance followed by DIK-SLAM and the traditional MCL algorithm. This implies that the RGBD-SLAM is able to cope with low feature environments better than other algorithms. The DIK-SLAM and MCL algorithm had the worst result because they are both particle-based algorithm that relies heavily on the feature quantities, as the features present in the environment increases, the better their performance, and in the absence of these features perform woefully [21]. Moreover, these outcomes were anticipated because the traditional MCL was revised to generate DIK-SLAM. Hence, the Freiburg2_360_kidnap dataset which is used to validate kidnapping and looping took a different course because the MCL technique and RTABMAP achieved a minimal ATE (RMSE) in their respective trajectories than both RGBD-SLAM and DIK-SLAM. Thus, this observation has led to an inquiry and thereafter comparing their trajectory to the ground truth, it was determined that the traditional MCL, the RTABMAP had been kidnapped earlier and were unable to recover. Therefore, limited ATE (RMSE) was estimated for their trajectories until the point they were kidnapped. Contrary, the DIK-SLAM and RGBD-SLAM, both algorithms completed the course of their trajectories and travelled further beyond others and were able to overcome kidnapping. Such accomplishment in RGBD-SLAM and DIK-SLAM also makes it possible to continue acquiring more ATE (RMSE) in their trajectories, since they are able to go beyond the point where the RTABMAP and MCL algorithms encountered kidnapping. Thus, for this reason, both DIK-SLAM and RGBD-SLAM achieved greater ATE (RMSE) in their trajectories than the RTABMAP algorithm and MCL. Also, it was observed that the margin of the ATE (RMSE) for both DIK-SLAM and RGBD-SLAM in the Freiburg2_360_kidnap experiment is the greatest in comparison to other environment scenarios. This is due to the frequent incidence of kidnapping and after recovery there is a considerable error arises during the position estimate and the build-up supports why the ATE (RMSE) is higher in Freiburg2_360_kidnap dataset for both algorithms compared to other environment scenarios. Furthermore, considering that both DIK-SLAM and RGBD-SLAM completed the course of their trajectory in Freiburg2_360_kidnap environment, the DIK-SLAM accomplished a lesser ATE(RMSE) as related to RGBD-SLAM algorithm. This suggests that during the kidnapping and looping condition, the DIK-SLAM performed better than other algorithms. The Freiburg3_sitting_static represents a static environment while the Freiburg3_long_office_household represents a large office room with visible structure and texture. In both datasets, the DIK-SLAM had the best performance with the lesser trajectory ATE (RMSE) while the RTABMAP held the second-best performance followed by RGBD-SLAM and the traditional MCL algorithm. Furthermore, all the SLAM algorithms attained the lowest ATE (RMSE) in the Freiburg3_sitting_static and Freiburg3_long_office_household datasets, compared to other environmental datasets because these datasets presented limited challenges. The environments offered a few cases of environmental noises, a few cases of overlapping sequences to create kidnapping and loop closure, static

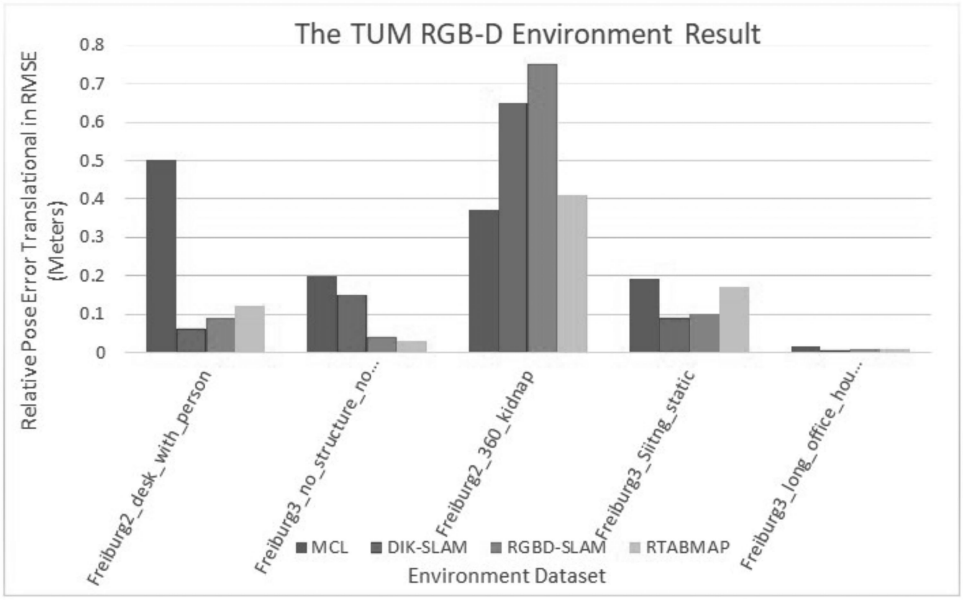

Figure 4. The relative pose error (RMSE) in meters.

objects with no complex features, such as corners, curves, *etc.* Instead, it is a flat surface with the visible known object. Thus, taking into consideration all datasets, it can be concluded that the DIK-SLAM outperformed other SLAM algorithms because it possesses lesser ATE (RMSE) than other SLAM algorithms except for the case of Freiburg2_360_kidnap where SLAM algorithms with lesser ATE (RMSE) than DIK-SLAM and RGBD-SLAM was because they got kidnapped earlier which resulted to limited ATE (RMSE) measurement. Likewise, in the case of Freiburg3_no_structure_no_texture_far, where the dataset contains a few samples to guide the robot trajectory resulted to limited the performance of DIK-SLAM.

4.1.1 Experiment 1.1: The Relative Pose Error (RPE) Translational (RMSE)

The datasets offered in Table 1 would be further analysed for position error based on the RPE. This RPE is a critical estimation that should be consider since it represents the drifting of the vehicle path from the real-life (ground truth) estimations [35]. This is necessary to evaluate the vision odometry technique. Hence, the RPE measure combines both translational and rotational error into a single metric, however, rotational errors are recorded indirectly by ATE. Since the two measures are so closely linked, the study's RPE estimate will be confined to the translational measurement (error) only. Equation (20) represents the RPE translational (RMSE) function employed to estimate the global error of trajectory at the time step k [35].

$$\mathrm{RMSE}\left(E_{1:m},\ \nabla\right) = \left(\frac{1}{n}\sum_{k\,=\,1}^{n} \left\|\mathrm{trans}\ \left(E_k\right)\right\|^2\right)^{\frac{1}{2}} \quad (20a)$$

where

$$n = m - \nabla, \quad (20b)$$

$\mathrm{trans}\left(E_k\right)$ denotes the translation elements of the relative pose error E_k, m denotes a sequence of camera poses, and n denotes each relative pose error per sequence [35]. The ∇ signifies a fixed time interval with an intuitive value set to 1. The deviation for each frame is estimated by $\mathrm{RMSE}\left(E_{1\,:\,\mathrm{m}}\right)$ as defined in (21) [35].

$$\mathrm{RMSE}\left(E_{1\,:\ \mathrm{m}}\right)\ =\ \frac{1}{m}\sum_{\nabla\,=\,1}^{m}\mathrm{RMSE}\left(E_{1:m},\ \nabla\right) \quad (21)$$

Thereafter the estimation of RPE for various environment scenes on an overall average based on a specific interval of time, the outcome is provided below (Fig. 4).

Considering the comparative experiment given in Fig. 4 between the algorithms. The MCL algorithm is depicted by a blue bar, the DIK-SLAM algorithm by a red bar, the RGBD-SLAM algorithm by a green bar, and the RTABMAP algorithm by a yellow bar. In the Freiburg2_desk_with_person environment, which suffers from a dynamic environment caused by the existence of moving items and environmental noise due to the presence of lighting variation and shadow. Thus, the result obtained has a similar behaviour to that of the ATE (RMSE) result presented in Fig. 3 with the DIK-SLAM performed best in terms of lower RPE (RMSE) in its trajectory. The RGBD-SLAM has the second-best performance as related to lower RPE (RMSE) followed by RTABMAP and the original MCL algorithm. This further support DIK-SLAM's potential to adapt to a dynamic situation and environmental noise better than other algorithms. In the Freiburg3_no_structure_no_texture_far which represents the environment with a low object. The result obtained has a contrary behaviour to that of the ATE (RMSE) presented in Fig. 3. This time around, the RTABMAP performed best in terms of lower RPE (RMSE) in its trajectory while the RGBD-SLAM has the second-best performance followed. Thus, the DIK-SLAM and MCL algorithm had

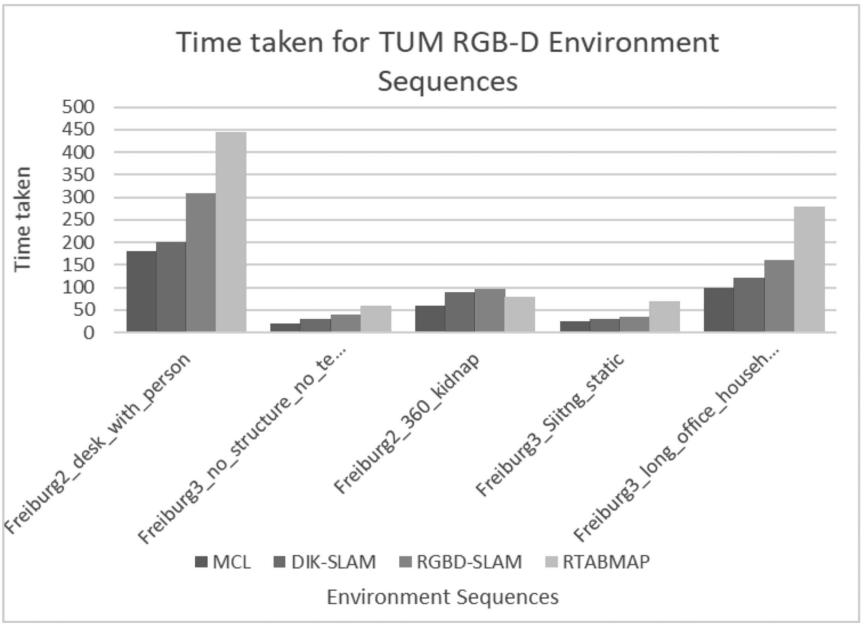

Figure 5. The time taken for TUM RGB-D environment sequences.

the least performance because they are particle-based algorithm that relies on object present in the environment to function perfectly and in the absence of these objects, their performance deteriorates, their performance is linked to the object's quantity present in its surroundings [31]–[33], [35]. In the Freiburg2_360_kidnap experiment which is used to validate kidnapping and looping, has a similar behaviour to that of the ATE (RMSE) result presented in Fig. 3. The MCL and RTABMAP have the lowest RPE (RMSE) in their trajectories compared to the DIK-SLAM and RGBD-SLAM. Further investigation shows that The MCL and RTABMAP got kidnapped earlier and could not recover and due to their limited trajectories, they are able to produce lower RPE (RMSE) up until the point of kidnapping. However, the DIK-SLAM and RGBD-SLAM were able to recover from kidnaping and completed the course of their trajectories and eventually are able to accumulate more RPE (RMSE) because they cover more grounds than the MCL and RTABMAP. Furthermore, comparison to other environment cases, the variance of the RPE (RMSE) for both DIK-SLAM and RGBD-SLAM in the Freiburg2_360_kidnap experiment is at its peak. This is because of regular incidence of kidnapping, and after recovering from all kidnap incidences, there is a significant error in its position estimation, and the build-up lead to a higher RPE (RMSE) in the Freiburg2_360_kidnap dataset for both techniques compared to other environment scenes. Given that both the DIK-SLAM and the RGBD-SLAM algorithms finished their trajectories in the Freiburg2_360_kidnap scenario, the DIK-SLAM produces a lesser RPE (RMSE) than the RGBD-SLAM algorithm. This demonstrates that the DIK-SLAM outperformed other algorithms during the kidnapping and looping conditions. The Freiburg3_sitting_static represents a static environment while the Freiburg3_long_office_household represents a large office room with visible structure and texture. In both datasets, there is a bit of twist in the experimental result when compared to Fig. 3, the DIK-SLAM maintained the best performance with the lowest RPE (RMSE), but the RGBD-SLAM now holds the second-best performance, followed by RTABMAP and the traditional MCL algorithm. Even though both datasets are not challenging, the Freiburg3_sitting_static dataset is not 100% static, its environment also contains slowly moving dynamic object which impacted the translational error and this explain why all algorithms attained higher RPE (RMSE) in the Freiburg3_sitting_static than the Freiburg3_long_office_household. Overall evaluation consider DIK-SLAM has the best SLAM algorithm because for most cases of the datasets, it attained lesser RPE (RMSE) than other SLAM algorithms. However, the case of Freiburg2_360_kidnap where SLAM algorithms attained a lesser RPE (RMSE) than DIK-SLAM and RGBD-SLAM was because they were kidnapped earlier, resulting in limited RPE measurements. Similarly, in the case of Freiburg3_no_structure_no_texture_far, where just a few samples are present in the dataset to guide the robot trajectory has limited the performance of DIK-SLAM.

4.2 Experiment 2: TUMRGB-D Dataset Processing Time Evaluation

The research considers computational cost as connected to processing speed, the MCL, DIK-SLAM, RGBD-SLAM, and the RTABMAP algorithms were further examined in this research by recording the time taken for processing each sequence of the environment. The performance result is presented in Fig. 5.

Given the outcome amongst the MCL depicted by a blue bar, the DIK-SLAM algorithm by a red bar, the RGBD-SLAM algorithm by a green bar, and the RTABMAP algorithm by a yellow bar as provided in Fig. 5. The observation is as follows: In Freiburg2_desk_with_people dataset which represents the

environment that suffers from dynamic and environmental noises shows that the MCL has the least execution (process) time, this means the speed of processing in MCL algorithm is the fastest, compared to other algorithms. Further investigation to understand why DIK-SLAM, RGBD-SLAM, and RTABMAP have higher processing times, reveal that they all are more computationally intense than the MCL algorithm and this caused slower processing speed which leads to higher processing time [21]. The Freiburg3_no_structure_no_texture_far represents the environment with a low object, the Freiburg3_sitting_static represents a static environment and the Freiburg3_long_office_household represents a large office room with visible structure and texture. Their performance behaviour is similar to that of the result obtained in the Freiburg2_desk_with_people, where the MCL-SLAM attained the most impressive processing speed, followed by the DIK-SLAM, RGBD-SLAM, and RTABMAP, respectively. Although, the processing time taken for processing these four datasets defers from each other. Given the amount of computational intensity and the processing time difference between DIK-SLAM and traditional MCL SLAM, the DIK-SLAM performance is acceptable and has the second lowest processing time. This accomplishment supports the effectiveness of the concurrency technique used to speed up the processing speed. The RGBD-SLAM holds the third best performance with a shorter processing time compared to RTABMAP with the worst processing time. Thus, in the Freiburg2_360_kidnap scenario, the observation behaviour is contrary to other environmental datasets. The MCL and RTABMAP algorithm has faster processing speed with lower processing time than the DIK-SLAM and RGBD-SLAM. Thus, there is a twist that complicated the comparison to acknowledge the algorithm with the best performance. The MCL and RTABMAP algorithm got kidnapped earlier in their navigation and the processing time was the only measure to the point of kidnapping whereas, the DIK-SLAM and RGBD-SLAM completed the course of their trajectory so they were able to acquire more processing time because they were able to recover from kidnapping. However, considering that the DIK-SLAM and the RGBD-SLAM algorithms finished the course of their trajectories in the Freiburg2_360_kidnap scenario and considering the computational complexity has linked to the execution (process) time. The DIK-SLAM algorithm achieved a lower processing time than the RGBD-SLAM algorithm. This implies that the DIK-SLAM even though we have increased its computational complexity attained a faster processing speed than the RGBD-SLAM, this observation is supporting the effectiveness of the concurrency technique (Rotating Tilling). Furthermore, in Fig. 5, we also observed that the level of the processing time for all algorithms in the Freiburg2_desk_with_people is relatively high compared to other environmental scenes. The investigation discovered that the size of the file's is huge as related to other environmental scenarios. The data size impacts the execution time, and because the file size is large, it requires a longer duration of time to be processed compared to other datasets [30]. The Freiburg3_long_office_household dataset size is the second largest and it requires a longer processing time for all the SLAM algorithm than the Freiburg3_no_structure_no_texture_far, Freiburg3_sitting_static, and Freiburg2_360_kidnap dataset.

5. Conclusion and Future Work

SLAM has become incredibly common because of its potential to support SLAM processes. This is an important step in overcoming the problem of creating an autonomous vehicle that can function independently with no human intervention. Unfortunately, due to the various limitations affecting SLAM techniques, this achievement has not been completely realised. Considering the literature review conducted in this study, factors, such as shadow, light intensity, kidnapping, loop closure, and dynamic environment continue to hinder the progress in SLAM [2], [5], [29], [31]–[33]. The proposed DIK-SLAM has been enhanced with filters operating concurrently to enable the SLAM algorithm to deal with the problem of environmental noise and slow processing speed. The DIK-SLAM approach was also modified to deal with dynamic environments, kidnapped vehicles, and looping closure by applying the cell occupancy technique and initialising localisation technique. The ATE based on root mean squared error was utilised to verify the vehicle trajectory, while MATLAB was used for simulation. The graphical results/findings showed that the DIK-SLAM on most occasions has better performance than the SLAM algorithms used in our assessment when compared with public datasets. Although, the study has accomplished so much but there are issues that need to be mentioned, and some of them will be attempted in our future work. Considering the study's goal of lowering computing costs, we were unable to suggest incorporating the Multi Tracking Object and Detection algorithm into DIK-SLAM [35]. This will allow us to track dynamic object moving around the environment unlike the present technique this is only detecting dynamic environment. Furthermore, the cell occupancy approach, which is founded on the idea that cells can adjust independently was presented in this research since it performs better with low computation complexity [35]. Thus, the hypothesis may not function in all scenarios since a group of nearby cells may be filled with the same item which contradicts the hypothesis, and in such instances, cells may become dependent on one another, with an effect that reduces system performance of the current SLAM system proposed. The most difficult problem is attaining 100 percent accuracy in trajectory under any conditions and it's not being attained at the present. Hence, there continues to be a demand for more research in SLAM.

References

[1] S. Ling-feng, F. Zheng, and Y. Shi, Multi-constraint SLAM optimisation algorithm for indoor scenes, *International Journal of Robotics and Automation, 38*, 2023, 1925–7090, ISSN (Online).

[2] C. Guo, K. Huang, Y. Luo, H. Zhang, and W. Zuo, Object-oriented semantic mapping and dynamic optimization a mobile

robot, *International Journal of Robotics and Automation, 37*, 2022, 1925–7090, ISSN (Online).
[3] C. Liu, G. Zhang, Y. Rong, W. Shao, J. Meng, G. Li, Y. Huang, Hybrid metric-feature mapping based on camera and Lidar sensor fusion, *Elsevier: Measurement*, *207*, 2023.
[4] V. Bianchi, P. Ciampolini, and I.D. Munari, RSSI-based indoor localization and identification for ZigBee wireless sensor networks in smart homes, *IEEE Transactions on Instrumentation and Measurement*, *68*(2), 2019, 566–575.
[5] S. Jia, K. Wang, and X. Li, Mobile robot simultaneous localization and mapping based on a monocular camera, *Journal Robot*, *2*, 2016, 7630340.
[6] Y. Dai, J. Wu, and D. Wang, A review of common techniques for visual simultaneous localization and mapping, *Hindawi Journal of Robotics, 20*, 2023, 1–21.
[7] S. Zheng, J. Wang, C. Rizos, W. Ding, and A. El-Mowafy, Simultaneous localization and mapping (SLAM) for autonomous driving: Concept and analysis, *MDPI Remote Sensing, 15*, 2023, 1–42.
[8] P.K. Panigrahi and S.K. Bisoy, Localization strategies for autonomous mobile robots: A review, *Journal of King Saud University: Computer and Information Sciences, 34*, 2022, 6019–6039.
[9] L. Paya, A. Gil, and O. Reinoso, A state-of-the-art review on mapping and localization of mobile robots using omnidirectional vision sensors, *Journal of Sensors (Hindawi), 20*, 2018, 3497650.
[10] I. Bukhori and Z.H Ismail, Detection of kidnapped robot problem in Monte-Carlo localization based on the natural displacement of the robot, *International Journal of Advanced Robotic Systems*, 2017, 1–6.
[11] R. Möller, A. Furnari, S. Battiato, A. Härmä, and G. M. Farinella, A survey on human-aware robot navigation, *Robotics and Autonomous Systems (Elsevier*), *145*, 2021, 103837.
[12] H. Cho, E.K. Kim, E. Jang, and S. Kim, Improved positioning method for magnetic encoder type AGV using extended Kalman filter and encoder compensation method, *International Journal of Control, Automation and Systems, 15*, 2017, 1844–1856.
[13] C.F. Lin, Y.C. Chung, and D.L. Yang, Efficient parallel volume rendering methods on distributed memory multicomputers, *Department of Engineering and Computer Science, 407*, 2003.
[14] J. Sarton, Y. Rémion, and L. Lucas, Distributed out-of-core approach for in-situ volume rendering of massive dataset, *International Conference on High Performance Computing, 11887*, 2019, 623–633.
[15] S. Wen, Z. Wang, J. Chen, L. Manfredi, and Y. Tong, CSLAM system consensus estimation in dynamic communication networks, *International Journal of Robotics and Automation, 37*, 2022, 1925–7090, ISSN (Online).
[16] J.J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J.M. Rendón-Mancha, Visual simultaneous localization and mapping: A survey, *Artificial Intelligence Review*, *43*, 2015, 55–81.
[17] A. Deeb, Y.J. Pan, and M.L. Seto, Model-based quasi-static SLAM in unstructured dynamic environments, *Proc. IEEE 17th International Conf. on Automation Science and Engineering (CASE)*, Lyon, 2021, 196–201.
[18] W. Shiguang, Y. Mingde, W. Chengdong, and L. Jun, An improved FastSLAM2.0 algorithm based on ant colony optimization, *Proc. 29th Chinese Control and Decision Conf. (CCDC),* Chongqing, 2017, 7134–7137.
[19] S. Badalkhani, R. Havangi, and M. Farshad, An improved simultaneous localization and mapping for dynamic environments, *International Journal of Robotics and Automation, 36*, 2021, 1925–7090, ISSN (Online).
[20] R. Guyonneau, S. Lagrange, L. Hardoun, and P. Lucidarme, The kidnapping problem of mobile robots: A set membership approach, *Proc. 7th National Conf. on Control Architectures of Robots*, 2014.
[21] Y. Tian and S. Ma, Kidnapping detection and recognition in previous unknown environment, *Journal of Sensors*, *2017*, 2017, 1–15
[22] Z.H. Ismail and I. Bukhori, Efficient detection of robot kidnapping in range finder-based indoor localization using quasi-standardized 2D dynamic time warping, *MDPI: Applied Science, 11*(4), 2021, 1580.
[23] S. Fuhrmann, F. Langguth, N. Moehrle, M. Waechter, and M. Goesele, MVE-An image-based reconstruction environment, *Computers & Graphics (Elsevier), 53*, Part A, 2015, 44–53.
[24] A. Jebelli, H. Chaoui, A. Mahabadi, and B. Dhillon, Tracking and mapping system for an underwater vehicle in real position using sonar system, *International Journal of Robotics and Automation, 37*2022, 1925–7090, ISSN (Online).
[25] H. Zhang and M. Li, Rapid path planning algorithm for mobile robot in dynamic environment*, Advances in Mechanical Engineering, 9*, 2017.
[26] Y. Olusanya Agunbiade, M. Seleman Ngwira, and T. Zuva, Enhancement optimization of drivable terrain detection system for autonomous robots in light intensity scenario, *International Journal of Advance Manufacturing Technology, 74*(5–8), 2014, 629–636.
[27] O. Guclu and A.B. Can, Fast and effective loop closure detection to improve SLAM performance*, Journal of Intelligent & Robotic Systems, 93*(5), 2019, 495–517.
[28] A. Kaser, Benchmarking and comparing popular visual slam algorithms, *Asian Journal of Convergence in Technology, 5*(1), 2019, 1–7.
[29] Y. Olusanya Agunbiade and T. Zuva, Image enhancement from illumination variation to improve the performance of simultaneous localization and mapping technique, *Proc. 4th International Conf. on Information and Computer Technologies (ICICT). IEEE Explore Digital Library*, HI, USA, 2021, 115–121.
[30] R. Baeza-yates and Z. Liaghat, Quality-efficiency trade-offs in machine learning for text processing, *Proc. IEEE International Conf. on Big Data (Big Data)*, Boston, MA, USA, 2017, 897–904.
[31] O.Y. Agunbiade and T. Zuva, A review: Simultaneous localization and mapping in application to autonomous robot, 2018, 2018050293 (Preprints).
[32] O.Y. Agunbiade and T. Zuva, Simultaneous localization and mapping in application to autonomous robot, *Proc. International Conf. on Intelligent and Innovative Computing Applications (ICONIC*), Idukki, 2018, 1–5.
[33] Y.O. Agunbiade, J.O. Dehinbo, T. Zuva, and A.K. Akanbi. Road detection technique using filters with application to autonomous driving system, 2018, *arXiv:1809.05878*.
[34] J. Kelsey, Timer functions in MATLAB: TIC and TOC functions, MATLAB Central File Exchange. https://www.mathworks.com/matlabcentral/fileexchange/106940-timer-functions-in-matlab-tic-and-toc-functions (retrieved 25, 2023).
[35] O.Y. Agunbiade, *Simultaneous localization and mapping for autonomous robot navigation in a dynamic noisy environment*, Ph.D. Thesis, Vaal University of the Technology, South Africa, 2022.

Biographies

Olusanya Y. Agunbiade obtained his Ph.D. degree from Vaal University of Technology, Faculty of Applied Science and Computer Science. He is currently a Senior Lecturer with the Department of Information Systems, University of the Western-Cape. He is an Active Researcher as well as a Postgraduate Supervisor for students in the faculty of ICT. His research interest covers a wide range of areas such as image processing, security, computer vision, recommender systems, computational intelligence, robotics, sensors, pattern recognition, computational intelligence, artificial intelligence, unmanned vehicle, simultaneous localization and mapping, biometric, cloud computing, intelligent system, information systems and management.

Mziwoxolo Mayedwa is a Lecturer with the University of the Western Cape. He specializes in information and communication technologies for development (ICT4D), robotics, sensors, pattern recognition, computational intelligence, artificial intelligence, and unmanned vehicle. He teaches information systems subjects and supervises post-graduate students within the information systems profession.

Awosejo Oluwaseun Johnson received the master's degree in business information systems from Tshwane University of Technology, South Africa. Then, he obtained an NRF & TUT scholarship for his D-Tech (Business Administration). He is well-known in South Africa and Nigeria for his outstanding innovative research work on the Internet of Things and artificial intelligence. Despite having worked in the industry as a Business Analyst and System Analyst, he enjoys working in the classroom, instructing students, developing projects, and sharing his skills and expertise with undergraduate and post-graduate students. He has been a Researcher with the Department of Informatics and Management Science since 2014. He began his academic career after earning a Higher National Diploma in Accounting from the Polytechnic of Ibadan in Nigeria.