A Comparative Analysis of Visual SLAM and LiDAR-based SLAM algorithms for Autonomous Vehicles

Sarvesh Bhalchandra Telang
Commercial Vehicle Technology
Technical University Kaiserslautern
Kaiserslautern, Germany
telang@rhrk.uni-kl.de

***Abstract*— SLAM has long been a prominent subject in the field of autonomous vehicles. Despite the fact that highly reliable technologies and solutions have developed over years of study and research, the SLAM problem is still regarded as unresolved. The two most popular SLAM techniques for autonomous navigation- Visual SLAM and LiDAR SLAM, are constantly debated. This literature provides a comprehensive analysis of the aforementioned techniques by evaluating the performance of each algorithm using existing experiments. Each algorithm has certain advantages and disadvantages because of the dynamic environment and sensor limitations. Bundle adjustment based vSLAM techniques such as ORB-SLAM, PTAM, and the graph optimization-based LiDAR method appear to be more reliable than other techniques. While visual sensors are less expensive and provide semantic information, LiDAR is highly mature and accurate sensor technology. Combining their capabilities would increase the robustness and resistance to adverse weather and lighting conditions and its ability to handle complex multi-modal problems.**

Keywords*— SLAM, Localization, LiDAR, ORB-SLAM

I. INTRODUCTION

Simultaneous localization and mapping, or SLAM, has gained much attention in the autonomous navigation sector for the past 20 years. The primary requirement for any autonomous vehicle navigation system is to know the precise location of the ego vehicle itself and to have an exact perception of its surroundings [2]. SLAM is capable of simultaneously executing both of these requirements, namely localization and mapping [2]. A self-driving vehicle can create a map of its surroundings using SLAM, and it can then use that map to determine where it is in relation to that environment [1]. Researchers have performed several investigations to determine the most effective SLAM implementation [2]. Although, in theory, it is a comprehensive solution for autonomous navigation, several problems arise in practice [2]. To create a map, the entity must have sensors for perceiving and measuring the aspects of its surroundings [5]. Sensors can be either exteroceptive or proprioceptive [5]. Exteroceptive sensors perceive the environment outside the area of the vehicle's mechanics, whereas proprioceptive sensors keep track of the vehicle's state and parameters such as wheel speed, wheel position, and battery charge [34]. Some examples of exteroceptive sensors are sonar, range lasers, cameras, and global positioning systems (GPS) [5]. Laser sensors, which are expensive and have high precision, are commonly used in automotive applications to perceive the environment; however, gathering surface information about objects using laser sensors is difficult [14]. Visual sensors have recently gained popularity and are frequently used in SLAM systems due to their advantages of being compact, inexpensive, and easily available [14]. Furthermore, multiple sensors, such as 3D and 2D LIDARs, Stereo, RGB-D, and monocular cameras, are often employed in SLAM to improve accuracy and reduce prediction errors [2]. However, a universal solution to the SLAM problems has not yet been fully developed, as each method that employs the aforementioned sensors has its own benefits and drawbacks [2].

Visual-based SLAM or vSLAM and LIDAR-based SLAM are the two most widely used and reliable SLAM techniques in today's industry. These methods use Cameras and LiDARs as exteroceptive sensors, respectively. However, they are two very different sensors, each with distinct advantages and disadvantages [2]. For instance, cameras are employed to obtain a semantic interpretation of the scene but are ineffective in poor lighting conditions. On the other hand, laser scanners are crucial for obstacle identification and tracking but are sensitive to rain [2]. Many vSLAM algorithms fail when working in external or dynamic environments with too many or much less features [5]. In this case of insufficient feature points, LiDAR (Light Detection and Ranging) or 3D mapping using laser scanners will be more effective [5].

LiDAR is a prominent sensor for motion estimation and 3D mapping in robotics and autonomous navigation. It is an ideal sensor for measuring distance to the obstacles and creating a grid map that describes the structure and obstacles on the vehicle's path [7]. It was frequently used as the primary sensor in early SLAM research. EKF or Extended Kalman filter was utilized in [9] for the prediction of the robot's pose and orientation. At each stage of the algorithm, it provides an estimate and lowers the uncertainty using a probabilistic model [7]. However, the results weren't satisfactory because this method produces additional truncation errors in case of highly nonlinear systems, which could result in improper localization and mapping [7]. To address this, [10] proposed using Particle filter techniques since they can efficiently avoid the nonlinear issue, but doing so also introduces an issue of increase in computation requirements as the particle number increases. Hence, new approaches needed to be developed. Even though LiDAR has many applications, the method for registering LiDAR scans has not changed much in the last ten years [2]. The scan-matching method, followed by a graph optimization, is the only primary solution for LiDAR-based navigation [2]. In section IV, these two approaches will be discussed in more detail.

Visual SLAM, also known as vSLAM, refers to mapping that uses cameras as the only exteroceptive sensor [5]. It is more complicated than LiDAR SLAM because, while images contain much information, measuring distance with them is difficult. [7]. Additionally, cameras can only capture a limited amount of visual information compared to 360° LiDAR, which is frequently employed in autonomous navigation [4]. Many Visual-SLAM algorithms, as mentioned in [11], are believed to have originated from Mono-SLAM, which was proposed in 2007. It was built around tracking and mapping feature points, known as a feature-based approach [11]. Another approach is utilizing the whole image without feature extraction, known as Direct SLAM [2]. This method is useful for dealing with texture-less or feature-less surroundings. The vSLAM algorithms in this article are classified as feature-based approaches, direct SLAM approaches, and RGB-D camera-based approaches, which are further explained in section III. The five basic modules: Initialization, tracking, mapping, relocalization, and global map optimization, make up the foundation of vSLAM algorithms [4]. Features and characteristics of the vSLAM algorithm heavily depend on the techniques used because each vSLAM algorithm uses a different methodology for each module [4]. As a result, it is necessary to comprehend each module to know its functionality, benefits, and drawbacks.

This article aims to examine and compare various algorithms used in the aforementioned visual and LiDAR SLAM techniques and assess their performance based on the results of case studies. The purpose of this comparison is to identify the most suitable, robust, and efficient algorithm in the field of autonomous vehicles. These techniques are compared across multiple operating domains and with various sensors to demonstrate each one's ability to generalize across various environments and sensor configurations. The outline of the paper is as follows: Second section provides a theoretical background for existing Visual SLAM algorithms such as feature-based SLAM, Direct methods, and RGB-D-based methods. Section 3 summarizes existing LiDAR algorithms and introduces scan-matching, graph optimization, and RBPF algorithms. In section 4, the general SLAM method is explained. Section 5 focuses on the experimental analysis to evaluate the discussed approaches using case study results. The outcomes of the same are covered in Section 6. The conclusion is finally provided in the last section.

II. Visual SLAM Algorithms

Basic Intro

A. Feature based approach

Feature-based approaches can be further classified into filtering-based (MonoSLAM) and keyframe or BA (Bundle adjustment)-based (PTAM and ORB-SLAM) methods [4]. These methods are based on the core principle of splitting the overall problem of generating geometric data from images into two separate processes [22]. The first process encompasses extracting a series of feature observations from the image, followed by computing the camera position and scene geometry solely based on these feature observations [22]. The following are some feature-based methods that serve as the foundation for visual SLAM algorithms:

1) MonoSLAM: In 2003, [12] developed the first successful MonoSLAM algorithm in the computer vision field, employing a single camera as an exteroceptive sensor [2]. It is a key approach in filter-based Visual SLAM techniques. In this method, EKF is used to estimate both camera motion and the three dimensional view of an unknown area [4]. We start with a known-size target to provide a precise scale for the predicted map and motion because there is no direct means of measuring feature depths or odometry due to the single camera [11]. In EKF, a state vector represents the 6 DoF motion of a camera and the 3D coordinates of feature points [4]. According to [11], the state vector $\hat{x}$ is made up of the stacked state estimations from the camera and features, and P is a square matrix with equal size that is decomposed into the submatrix elements shown in equation (1).

$$\hat{x} = \begin{pmatrix} \hat{x}_v \\ \hat{y}_1 \\ \hat{y}_2 \\ \vdots \end{pmatrix}, \; P = \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} & \cdots \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_2} & \cdots \\ P_{y_2x} & P_{y_2y_1} & P_{y_2y_2} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}. \tag{1}$$

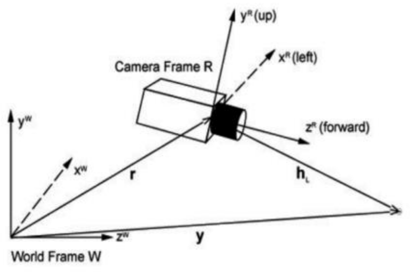

Fig.1: Camera and world frames [12, Fig. 1]

The coordinate systems for the camera and the world are shown in Fig. 1 [12, Fig. 1], where we define the coordinate frame R, fixed with regard to the camera, and frame W, fixed in the world [12]. y_i denotes feature states, which are 3D position vectors indicating the co-ordinates of point features [11]. The starting camera position in the state vector is given some level of uncertainty, up to a few degrees and centimeters [11]. Then, based on camera movement, this vector is updated with new feature points [4]. As stated in [11], feature point matching can be achieved using rectangular patches, as illustrated in Fig. 2 [11, Fig. 1(b)]. As too many feature points must be saved in the state, such filter-based approaches have been demonstrated to have limitations in the event of a large environment [2].

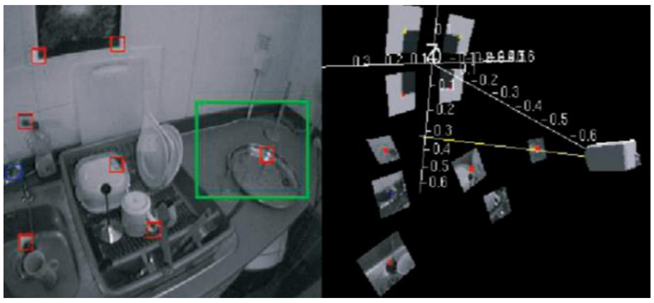

Fig. 2 MonoSLAM: The left image shows planar patches (landmarks) in rectangles, while the right image depicts the estimation of its position and camera pose [11, Fig. 1(b)].

Because there are too many points in a large environment, the vector grows in size, increasing the computational cost and making real-time calculation difficult [4]. To address this issue of high computing cost, [13] presented the PTAM method in 2007, which will be discussed further [2].

2) PTAM: Parallel Tracking and Mapping is referred to as PTAM. In 2007, [13] proposed splitting tracking and mapping into 2 distinct processes. For the first time, the concept of separating the front end and back end was introduced, setting the stage for the design of several SLAM techniques [7]. Additionally, it was the first technique to integrate Bundle Adjustment (BA) into real-time visual SLAM algorithms [4]. At first, the map is initialized with a large number of points from a stereo pair of cameras using five-point algorithm [13]. Then, in tracking, 2D-3D correspondences are generated by projecting points onto an image, from which camera poses can be obtained [4]. After that, triangulation is employed to calculate the coordinates of 3D feature points, and BA is applied to optimize those positions [4]. The PTAM example is shown in Fig. 2 [13, Fig. 1], where the online map has around 3000 points, and the algorithm tried to locate 100 of them in the current frame.

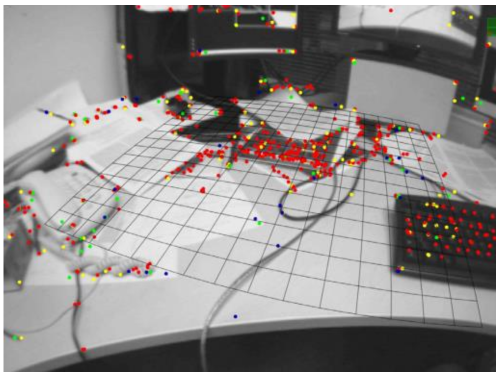

Fig. 2: PTAM [13, Fig. 1]

PTAM was successful in solving a problem of computational cost in MonoSLAM; however, the local minimum problem in BA makes it challenging to get global optimal camera and map poses in large environments [4]. Closed-loop detection and pose-graph optimization methods can be employed prior to bundle adjustment to prevent this issue [4]. These methods are used in ORB-SLAM, a BA-based algorithm that is an extended version of PTAM [4], [17]. The following section will go into detail about ORB-SLAM.

3) ORB SLAM: ORBs are referred to as oriented multiscale fast corners with a 256-bit descriptor, as mentioned in [17] and [18]. They offer strong perspective invariance and are quick to compute and match [17]. The ORB-SLAM employs three concurrent methods: the first one is tracking, the second is local mapping, and the final one is loop closing [17]. Here, tracking is responsible for localizing the camera and deciding when to add a new keyframe for each frame [17]. Figure 3 [17, Fig.1] provides a complete overview of this process.

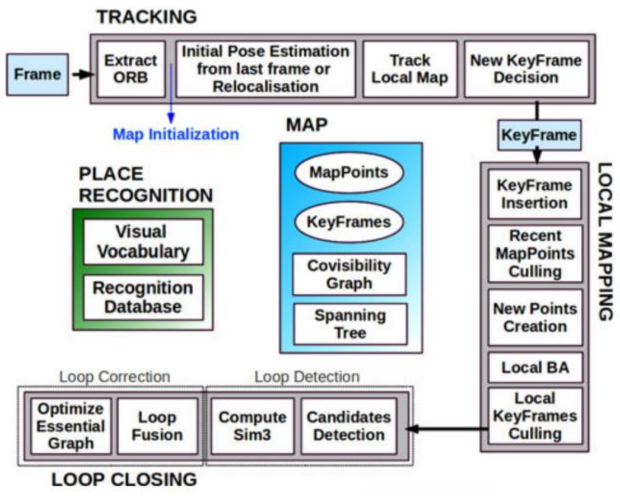

Fig. 3: Overview of ORB-SLAM algorithm displaying all the required stages [17, Fig. 1]

This is the most comprehensive and accurate feature-based approach available [4]. This technique can handle monocular, stereo, and RGB-D camera combinations with various algorithms and includes the majority of techniques to boost SLAM's performance [2]. The issue with this algorithm is that there are a lot of input parameters that need to be adjusted in order for the SLAM to function in a particular environment [2]. ORB-SLAM can be further expanded to include stereo vSLAM and RGB-D vSLAM [19].

B. Direct SLAM Approach

Direct techniques or feature-less approaches, in contrast to feature-based methods in the preceding section, utilize an input image with no abstraction [4]. Unlike feature-based approaches that use geometric consistency to measure error, direct methods use photometric consistency for error measurements [4]. A few well-known direct techniques are introduced in this section.

1) DTAM: In 2011, [20] proposed a straightforward technique called direct tracking and mapping. In this method, the input image is compared to synthetic images produced from the rebuilt map to perform tracking [4]. To track camera movements at specific frame-rate, whole image alignment is utilized from the dense model of a scene [20]. When the previous predicted image has fewer pixels than a certain threshold, a new keyframe is inserted [20]. This is feasible because the system is fully dense, and theoretically more well-structured than the feature-based systems [20].

2) LSD-SLAM: The fundamental concept of LSD-SLAM (Large-Scale Direct Monocular SLAM) is a technique similar to semi-dense visual odometry [21]. LSD-SLAM enables for the generation of accurate, comprehensive maps of the surroundings (as shown in Fig. 4 [22, Fig. 1]) in addition to tracking the camera's movement locally [22]. Unlike DTAM, which reconstructs entire landscapes, this method's reconstruction goals are restricted to areas with intensity gradients [4]. This implies that it ignores areas without textures because it is challenging to infer precise depth data from images [4].

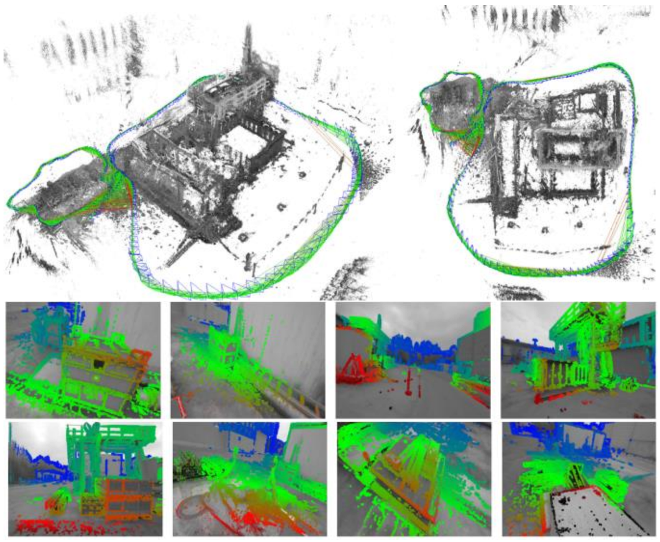

Fig. 4: LSD-SLAM- Top image: Real-time point clouds of complete keyframes of medium size trajectory, Bottom images: Selected keyframes in the depth map [22, Fig. 1]

The map provides a highly precise and semi-dense three-dimensional reconstruction of the surroundings in the form of point clouds [22]. According to the results of [22], this method successfully tracks and maps difficult hand-held trajectories with over 500 m distances, particularly those with large-scale variations (mean inverse depths ranging from less than 20 cm to more than 10 m) and big rotations, proving its adaptability, robustness, and flexibility [22].

C. *RGB-D vSLAM*

RGB-D structured light camera sensors [23] such as Microsoft Kinect fusion [24] and SLAM++ [25] have lately become more compact and affordable [2]. These cameras can provide both depth and color 3D data in real time, but because their range is only 4 to 5 meters and the technique is highly sensitive to sunlight, they are most likely used for indoor navigation [2]. As a result, they are ineffective in dynamic outdoor fields such as autonomous vehicles.

III. LiDAR SLAM Algorithms

LiDAR, or 3D mapping using laser scanners, is still a widely used technology in the field of autonomous vehicles due to its ease of use and accuracy [2]. This section will provide theoretical background on three reliable Lidar-based methods: scan-matching, graph optimization and particle filter based SLAM method RBPF (Rao-Blackwellized particle filters).

A. *Scan-matching approach*

Scan matching is the key step in creating 3D maps with LiDAR that provide accurate information on motion and it is mainly is based on the Iterative Closest Point (ICP) [2]. Using this technique, one can determine which point on a geometric shape is closest to a given point [27]. First, the correspondence, or degree of proximity between the two scans in terms of Euclidean distance is calculated [27]. If r_i is the reference scan and c_i is the current scan, the point of correspondence y_i can be calculated using equation (2) [27].

$$y_i = \arg_{r_i} \, min\, \{\| \, r_i - c_i \, \|\} \tag{2}$$

Second, according to [27], a transformation $T = [\phi_0, T_x, T_y]$ that minimizes the Euclidean distance between points is computed as follows:

$$T = \arg_T \min\{\textstyle\sum_i \| \, T.c_i - p_i \, \|\} \tag{3}$$

The ICP algorithm finds a convergent transformation between two scans by iteratively repeating these two processes [27]. The key drawbacks of ICP are its intensive search for point matching and its high sensitivity for minimizing the starting point [2].

B. *Graph based optimization*

There are two aspects to a graph-based SLAM architecture: Front-end and Back-end, as seen in the Figure 5 below [7, Fig.1]. By using sensor data, the front-end primarily serves to estimate the vehicle's position and the back-end component reduces accumulated errors and enhances the positioning and map accuracy by using optimization methods [7].

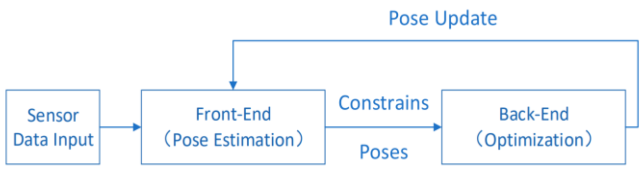

Fig. 5: A general framework for graph-based LiDAR-SLAM [7, Fig. 1]

By abstracting the unprocessed sensor measurements, it creates a simplified estimate issue [2]. The edges of the graph serve as virtual measures, replacing the actual measurements [2]. Figure 6 [2, Fig. 3] depicts the model of the process. Each node x_i in the network represents a different vehicle position [2]. The spatial restraints between the poses i and j are represented by the edges E_{ij} = $\{e_{ij}$, Ω_{ij} ij$\}$ which connect nearby poses [2]. The proprioceptive information constraint is described by edges e_{ij}, while the exteroceptive constraint is shown by edges Ω_{ij} [2].

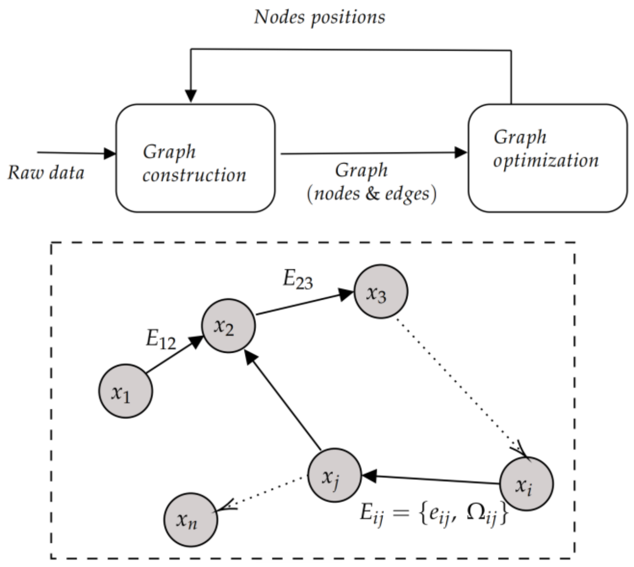

Fig. 6: SLAM pose-graph model [2, Fig. 3]

C. *RBPF*

According to [10], Rao-Blackwellized particle filters (RBPF) method makes use of a particle filter where every particle has a unique map of the surrounding environment. The complexity of this procedure is what makes it challenging because it takes so many particles to create an accurate map [10]. Also, the accurate particle can possibly be eliminated during the resampling process. In addition to proposing a method to compute a precise proposal distribution, [10] also introduced adaptive approaches for lowering the number of particle filters in an RBPF while also accounting for the most recent observation. The proposal distribution is calculated by computing the probability of a particle-dependent most-likely pose that was acquired using a scan-matching method and odometry data [10]. By using the last sensor observation to create the new generation of particles, it is possible to predict the ego vehicle's status using a model that is more informative and accurate than the one that was created using only odometry data [10].

IV. Slam Method

A. *Principle*

SLAM follows a recursive estimation approach, estimating both a variable M that represents the location of landmarks and a variable X that comprises the trajectory or posture of the vehicle [2]. According to [2], the following equation (4) can be derived given a set of measurements $Z = \{z_1, \ldots, z_m\}$ and a measurement or observation model h expressing z_k as a function of X and M:

$$z_k = h(X_k, M_k) + \epsilon_k \tag{4}$$

where, $\in_k$ stands for random noise in measurement, and X_k, M_k, respectively, signify a subset of X and M.
As mentioned in [2], the MAP problem (Maximum A Posteriori) is frequently resolved by SLAM in the following ways:

$$\begin{aligned}\{X^*,\ M^*\} &= argmax_{\{X,M\}}\ p(X,M|Z) \\ &= argmax_{\{X,M\}}\ p(Z|X,M)\ p(X,M)\end{aligned} \quad (5)$$

where, $p(X,M)$ represents a priori knowledge of X and M, while $p(Z|X,M)$ denotes the likelihood of the measurement Z [2]. If observations z_k are assumed to be independent, as stated in [2], the MAP equation is as follows:

$$\begin{aligned}\{X^*,\ M^*\} &= argmax_{\{X,M\}}\ \prod_{k=1}^{m} p(z_k|X,M)\ p(X,M) \\ &= argmax_{\{X,M\}}\ \prod_{k=1}^{m} p(z_k|X_k,M_k)\ p(X,M)\end{aligned} \quad (6)$$

B. *Solution of SLAM using probabilistic approach*

The SLAM process is frequently viewed probabilistically, requiring the traditional prediction and update steps [2]. According to [2], in case of a vehicle moving through an unknown area, we can define following parameters:

x_k : The state vector of the vehicle at time k
$x_{k|k-1}$: The predicted state vector at time k given the preceding state
u_k: Control vector (for the movement of vehicle from x_k to $x_{k|k-1}$)
m_i: Vector representing the landmark at i^{th} position
$z_{k,i}$: i^{th} Observation
X : Vehicle positions set for time 0 to k
$U_{0:k}$: Control input set for time 0 to k
$Z_{0:k}$: Observations set for time 0 to k
M : Landmarks set
$M_{k|k-1}$: The predicted map at time k knowing given the preceding map

As mentioned in [2,] the probability distribution function at every time k can be calculated using the following equation:

$$P(x_k, M|Z_{0:k}, U_{0:k}) \quad (7)$$

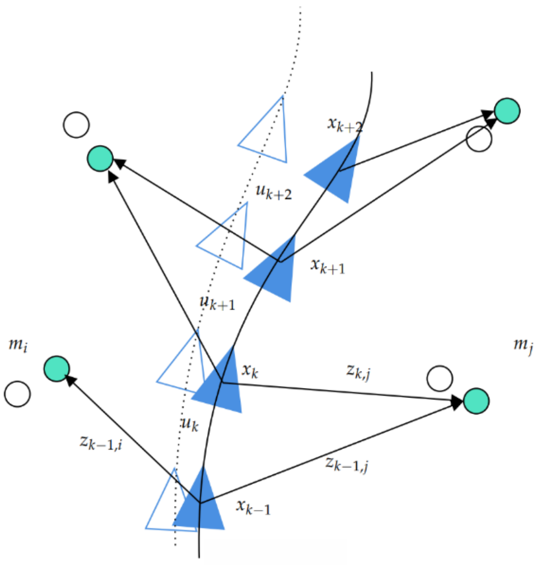

Fig. 7: Illustration of the general SLAM technique

To continue, we must employ an iterative solution that updates equation $P(x_{k-1|k-1}, M_{k-1|k-1}|Z_{0:k-1}, U_{0:k-1})$ with u_k and z_k [2]. To do this, as stated in [2], we must first develop a motion model that predicts the state provided the control input $P(x_k|\ x_{k-1}, u_k)$ in the following way:

$$\begin{aligned}&P(x_{k|k-1}, M_{k|k-1}|Z_{0:k-1}, U_{0:k}) = \\ &\int P(x_{k|k-1}|\ x_{k-1|k-1}, u_k) * \\ &P(x_{k-1|k-1}, M_{k-1|k-1}|Z_{0:k-1}, U_{0:k-1})dx_{k-1|k-1}\end{aligned} \quad (8)$$

This equation (8) provides the iterative Bayesian technique for SLAM implementation [2]. All the variables utilized in this section are represented by the SLAM process illustrated in Figure 7 [2, Fig. 2] below. The blue triangles in this figure represent the predicted trajectory of the vehicle, while the white triangles represent the ground truth, or the actual vehicle path. To efficiently compute the recursive technique, a SLAM solution must perform a suitable computation for both the motion and perception model [2]. Modern methods frequently make a prediction regarding the movement of the vehicle using inertial sensors [2]. The final observation model, as stated [2], is given in the equation (9) as follows:

$$P2D = \prod(\ P3D, K, T) \quad (9)$$

where, $T = [R, t]$ represents the rigid transform, K represents sensor's intrinsics.
Similarly, according to [2] the equation for solving SLAM using a graph-based approach is,

$$P3D_{map} = T(P3D_{LiDAR}) \quad (9)$$

In the case of LiDAR sensors, as shown in equation (9), a classical rigid transform has directly connected the observation to the state [2].

V. Experimental Analysis

A. *Evaluation of LiDAR based approaches*

The analysis presented here is meant to demonstrate the properties of the above mentioned techniques. This evaluation takes into account the case study from [26] and compares three techniques in-depth: Scan matching, RBPF, and graph optimization. The experiment in [26] is carried out in an indoor environment using LiDAR sensors for a holonomic robot. In the case of outdoor applications like autonomous vehicles, there is a dynamic environment with many obstacles and measurement errors. However, the procedure is the same, and it can be applied similarly in the case of a non-holonomic vehicles. Because, as stated in [29], the results for LiDAR sensors do not differ significantly between indoor and outdoor scans except for extreme weather conditions. [26] "selected the dataset of building 079 at the University of Freiburg, the Intel Research Lab dataset, and a dataset acquired at the CSAIL at MIT." [26, p.4] According to [26], initially an incremental scan-matching method is used as a baseline strategy, where the method of [28] is used to iteratively estimate an open loop likelihood trajectory for the vehicle. Then, the GMapping approach, a RBPF based mapping system, is implemented [26]. RBPF (Rao-Blackwellized Particle Filter) uses a particle filter to estimate the probability over maps and trajectories [26]. In the third step, a strategy that uses graph optimization to solve the SLAM problem is chosen. In this strategy, a graph is created using the measurements in a particular order and the error is minimized using leat square method to determine the most probable configuration [26]. The calculated error has two components, as shown in the Table I [26, Table I]: a translational error and a rotational error.

TABLE I [26, Table I]

RESULTS OF LIDAR-BASED TECHNIQUES

[1] DENOTES SCAN MATCHING IS USED AS A PREPROCESSING STEP.

Translational error m / m^2	Scan Matching	RBPF (50 part.)	Graph Mapping
Aces (abs) Aces (sqr)	0.173 ± 0.614 0.407 ± 2.726	0.060 ± 0.049 0.006 ± 0.011	0.044 ± 0.044 0.004 ± 0.009
Intel (abs) Intel (sqr)	0.220 ± 0.296 0.136 ± 0.277	0.070 ± 0.083 0.011 ± 0.034	0.031 ± 0.026 0.002 ± 0.004
MIT (abs) MIT (sqr)	1.651 ± 4.138 19.85 ± 59.84	0.122 ± 0.386[1] 0.164 ± 0.814[1]	0.050 ± 0.056 0.006 ± 0.029
CSAIL (abs) CSAIL (sqr)	0.106 ± 0.325 0.117 ± 0.728	0.049 ± 0.049[1] 0.005 ± 0.013[1]	0.004 ± 0.009 0.0001 ± 0.0005
FR 79 (abs) FR 79 (sqr)	0.258 ± 0.427 0.249 ± 0.687	0.061 ± 0.044[1] 0.006 ± 0.020[1]	0.056 ± 0.042 0.005 ± 0.011

Rotational error deg / deg^2	Scan Matching	RBPF (50 part.)	Graph Mapping
Aces (abs) Aces (swr)	1.2 ± 1.5 3.7 ± 10.7	1.2 ± 1.3 3.1 ± 7.0	0.4 ± 0.4 0.3 ± 0.8
Intel (abs) Intel (sqr)	1.7 ± 4.8 25.8 ± 170.9	3.0 ± 5.3 36.7 ± 187.7	1.3 ± 4.7 24.0 ± 166.1
MIT (abs) MIT (sqr)	2.3 ± 4.5 25.4 ± 65.0	0.8 ± 0.8[1] 0.9 ± 1.7[1]	0.5 ± 0.5 0.9 ± 0.9
CSAIL (abs) CSAIL (sqr)	1.4 ± 4.5 22.3± 111.3	0.6 ± 1.2[1] 1.9 ± 17.3[1]	0.05 ± 0.08 0.01 ± 0.04
FR 79 (abs) FR 79 (sqr)	1.7 ± 2.1 7.3 ± 14.5	0.6 ± 0.6[1] 0.7 ± 2.0[1]	0.6 ± 0.6 0.7 ± 1.7

Table I (top) [26, Table I] shows the translational error of the three methods, while Table I (bottom) [26, Table I] displays the rotational error. As can be observed, the RBPF and Graph-optimization methods have lower error values than the Scan-matching approach for both translational and rotational cases. Hence, it is clear that more advanced algorithms, such as the RBPF and graph mapping, are performing better than scan matching. The fundamental reason for this is because the result is optimized locally using scan matching and it will generate topological errors, particularly when big loops are needed to be closed [26]. The error values of Graph optimization and RBPF are very close. As a result, distinguishing between RBPF and graph mapping appears to be difficult because both techniques are performing well in this scenario. In terms of mapping, however, graph optimization method appears to be marginally superior to the RBPF.

Figure 6 [26, Fig. 4] provides additional insights into the metric by illustrating and visualizing the data with graphs indicating the error for each method. These graphs depict the results of an experiment carried out at the University of Freiburg's Building 079, which is a typical office building with each room connected by a single corridor [26]. It shows three columns: a graph-based method on the right, RBPF in the middle, and scan-matching in the first column. The area highlighted in blue in the first image is where the robot returns to a previously visited region. The error plots show that the errors are larger for the above 1000 relations [26]. The unclear map in the first image, as well as the fact that some walls appear twice, are additional indicators that the robot's posture estimate is inaccurate and that the inaccuracy accumulates [26]. The more advanced algorithms, such as RBPF and graph mapping, on the other hand, can create consistent and precise maps in this setting. Only a small minority of relations (shown by dark blue relations) exhibit an increased inaccuracy.

B. *Evaluation of Visual SLAM approaches*

1) Comparison between MonoSLAM (filter-based) and PTAM (BA-based) methods: As previously discussed in Section III with the theoretical background of various visual SLAM algorithms, feature-based algorithms are primarily either filter-based or BA-based. Using the findings of an experiment published in the literature [30], this section will explain how EKF-based (filter based) mapping in MonoSLAM differs from BA or keyframe-based mapping in

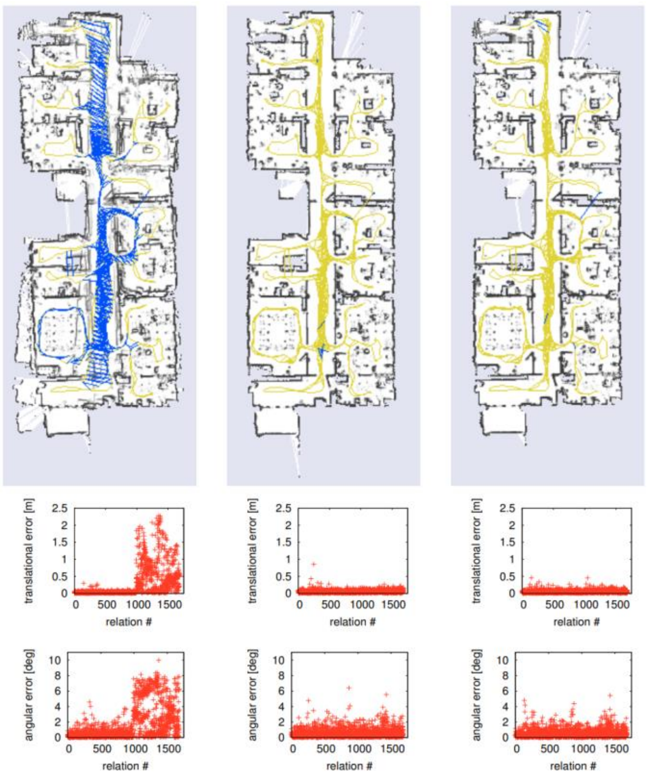

Fig. 6: Comparison of LiDAR based methods using the dataset of University of Freiburg Building 079. First column: Scan-matching result, Middle column: RBPF, Right column: Graph based optimization. The top image in each column depicts the map, the center plot represents the translational error, and the bottom plot represents the rotational error [26, Fig. 4]

PTAM. Using a combined cost and accuracy measure, [30] carried out a number of simulation-based experiments along with real-image visual SLAM system evaluations. The findings of the same are displayed in the following Fig. 7 [30, Fig. 10].

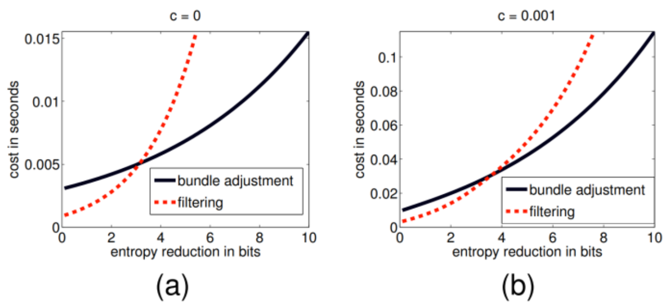

Fig. 7: Measurement of accuracy or cost in bits per second: Solid line indicates results of BA, while dotted line indicates the results of filtering based method [30, Fig. 10]

It can be seen that filter based techniques, such as EKF, are only effective at lower accuracy levels, or less than 3 bits. This holds true whenever the tracking cost is 1 (c = 1ms) or zero. This is the only area where Filtering-based MonoSLAM systems seem to be better than Bundle adjustment (BA) based methods. However, it is clear that BA is more effective for higher accuracy levels. This is mostly due to the fact that the cost of BA is linear in no. of features in contrast to the cubic cost of EKF [30]. These findings lead to the conclusion that adding more feature points to a map is necessary to increase the accuracy of vSLAM. The BA-based technique is superior to the EKF-based approach from this perspective since it can manage a large number of points.

2) Comparison between ORB-SLAM, PTAM, LSD-SLAM and RGB-D SLAM methods: Both ORB-SLAM and PTAM methods are BA-based, but they produce different results in practice. This section provides a detailed comparison between these BA-based and RGB-D based methods using the results of the experiment conducted by [17]. For this experiment, [17] utilized the TUM RGB-D benchmark dataset from [32], which offers a number of sequences with precise ground truth values.

TABLE II [17, Table III]
COMPARISON OF KEYFRAME LOCALIZATION ERRORS IN THE DATASET TUM RGB-D BENCHMARK [32]

	Absolute keyframe trajectory error (RMSE) in cm			
Data sets	ORB-SLAM	PTAM	LSD-SLAM	RGB-D SLAM
fr1_xyz	**0.90**	1.15	9.00	1.34 (1.34)
fr2_xyz	0.30	**0.20**	2.15	2.61 (1.42)
fr1_floor	**2.99**	X	38.07	3.51 (3.51)
fr1_desk	**1.69**	X	10.65	2.58 (2.52)
fr2_360 _kidnap	3.81	**2.63**	X	393.3 (100.5)
fr2_desk	**0.88**	X	4.57	9.50 (3.94)
fr3_long _office	**3.45**	X	38.53	–
fr3_nstr_tex_far	ambiguity detected	4.92 / 34.74	18.31	–
fr3_nstr_ tex_near	**1.39**	2.74	7.54	–
fr3_str_tex_far	**0.77**	0.93	7.95	–
fr3_str_ tex_near	1.58	**1.04**	X	–
fr2_desk_person	**0.63**	X	31.73	6.97 (2.00)
fr3_sit_xyz	**0.79**	0.83	7.73	–
fr3_sit_halfsph	**1.34**	X	5.87	–
fr3_walk_xyz	**1.24**	X	12.44	–
fr3_walk_halfsph	**1.74**	X	X	–

To compare these algorithms to the groundtruth, a similarity transformation is used to align the keyframe trajectories because the scale was unclear [17]. The results are shown in Table II [17, Table III].

With the exception of dataset fr3_nstr_ tex_ far, it is clear from the table that ORB-SLAM can handle all of the other sequences [17]. It shows that ORB-SLAM is capable to identify ambiguities. PTAM occasionally selects the wrong solution, which is unacceptable [17]. Although there is no ambiguity in the case of LSD-SLAM, the error seems extremely high compared to other methods. PTAM and LSD-SLAM are less robust than ORB-SLAM in the remaining sequences, losing track in total 8 and 3 sequences, respectively [17]. Open trajectory accuracy for ORB-SLAM and PTAM is similar, but as seen in the sequence fr3_nstr_tex_near, ORB-SLAM performs better when identifying huge loops [17]. The most unexpected finding is that LSD-SLAM and RGBD-SLAM are outperformed by PTAM and ORB-SLAM in terms of accuracy [17]. One of the likely explanations for this is that they convert the map optimization into a graph optimization, discarding the sensor readings as we conduct BA [17].

C. *Comparison of Visual vs LiDAR SLAM approaches*

As individual algorithms in visual and LiDAR-based SLAM have already been evaluated in previous sections, this section will focus on a thorough comparison of the two main techniques. This analysis considers the article [15], in which a comparative experiment between PTAM and PF-SLAM is carried out to examine the localization accuracy. Please note that, as previously stated, PTAM is a BA-based Visual SLAM method, whereas PF-SLAM is a particle filter-based LiDAR SLAM method. Section IV has already covered the RBPF technique, an extension of the PF-SLAM. In the experiment described in [15], which is conducted in the test field shown in Fig. 8 [15, Fig. 8 a)], the robot is given instructions to move in a round trajectory with a radius of 2.3 m [15]. The robot is equipped with a LRF Sensor (laser rangefinder) [15]. The PF-SLAM and PTAM are run simultaneously after stereo initialization of PTAM, and measurement data is recorded in the same setting [15].

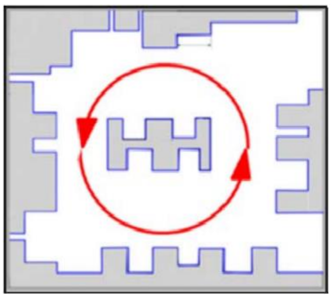
Fig. 8: Robot trajectory [15, Fig. 8 b)]

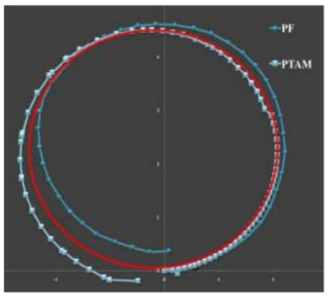

Fig. 9: Result of PF-SLAM and PTAM [15, Fig. 10]

With 14000 features and 220 keyframes, the PTAM generated a fairly dense map [15]. The result of the PTAM comparison with PF is shown in Figure 9 [15, Fig. 10]. The paths of ground truth are represented by the red line. As seen in the figure, PF-SLAM results deviate with time from the ground truth trajectories while PTAM localization results tend to approach the ground truth [15]. This shows that PTAM is more accurate than PF-SLAM. The primary spatial structure, as determined by a monocular camera, is shown in Fig. 10 [15, Fig. 12].

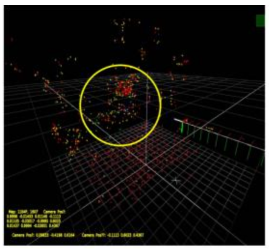
Fig. 10 PTAM feature points [15, Fig. 12]

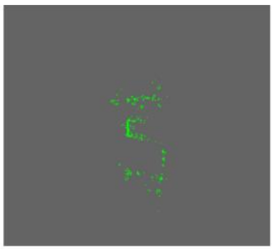
Fig. 11 Camera's global environment map [15, Fig. 13]

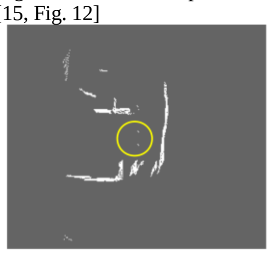
Fig. 12 LRF's global environment map [15, Fig. 14]

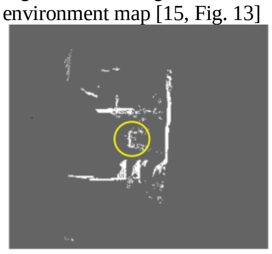
Fig. 13 Global environment map - Integration [15, Fig. 15]

PTAM has identified 2695 feature points and 15 keyframes [15]. Figure 11 [15, Fig. 13] shows a camera's global environment gird map that is created through the projection of feature points [15]. Fig.12 [15, Fig. 14] is created by combining the robot positions provided by the PTAM tracking thread with LRF [15]. Utilizing a monocular camera, LRF, and integration rule, Fig. 13 [15, Fig. 15] is created [15]. Because vision information is added and the resulting map often resembles the ground truth, the map is much refined [15].

To summarize the evaluation results, Bundle adjustment (BA) based approaches and Pose graph estimation methods produce more accurate results than filter based methods. While traditional scan matching and filter-based LiDAR methods fail to deliver satisfactory results, the pose graph optimization LiDAR-based method does provide accurate results. It also enables large-scale SLAM and eliminates the raw data from the optimization step, allowing for a rather simple loop closing process [2]. As a result, many self-driving cars today use a combination of LiDAR and camera sensors. This will be covered in the section that follows.

VI. Results and Discussion

Visual sensors have the benefit of currently being the subject of extensive research. Even though vSLAM yields precise results, there are some drawbacks, such as the drift of the scale in case of monocular cameras, poorly estimated depth or delay in initialization of depth, the limited field of view for stereo-vision, the sparseness of the reconstructed maps, and the challenge of using RGB-D in outdoor environments [2]. As seen before, the methods employed for LiDAR-based SLAM rely on scan matching and graph pose estimation. The primary benefit of LiDAR is its extremely high range accuracy, which also has a positive impact on mapping. It makes clear that combining both techniques would be extremely beneficial to the field of autonomous vehicles. However, using both modalities necessitates a challenging initial calibration procedure [2]. Precision calibration between the two sensors is required for good performance of SLAM with a LiDAR-camera fusion [2]. According to [2], to ascertain the relative transformation between the camera and the LiDAR, an extrinsic calibration is required as shown in the figure 10 [2, Fig. 5]. The task is to determine the rigid transform M_L^C between the camera and the

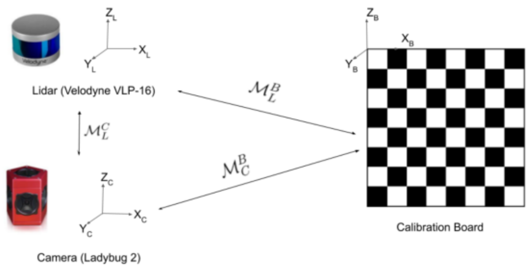

Fig. 10: Extrinsic calibration principle [2, Fig. 5]

LiDAR [2]. Currently, it is primarily carried out manually by identifying a calibration target, such as a 2D or 3D checkerboard or pattern, for each modality (M_L^B and M_C^B) [2]. In addition, a major issue with SLAM is loop detection. The cumulative error can be effectively reduced and the positioning accuracy can be increased by recognizing the previous locations and applying loop pose constraints to the pose graph [7]. Because scan data only describes the obstacle structure on the LiDAR installation surface, which typically lacks distinctive aspects of the scene [7]. LiDAR-based SLAM algorithms frequently fail to recognize the loop successfully [7]. To put it simply, different locations may have scan data that is very difficult to distinguish, such as long coordinators, similar looking vehicles and roads with similar layouts. Furthermore, LiDAR SLAM requires a lot more processing power because of the type of data it works with, despite being more accurate, precise, and prone to fewer errors. Second, the costs of infrastructure and the related hardware for LiDAR sensors are currently disproportionately high. Other perceptional tasks, such as signboard and object detection, are also far more difficult. This can be easily improved by incorporating visual texture data and scene feature points. Visual sensors essentially meet all of the criteria for use in self-driving vehicles. According to the acceptability criteria for sensors mentioned in [33], the sensor must be affordable in order to compete on the market, self-contained for easy deployment and use on vehicles, and easily mountable and demountable in order to be deployed in the field by non-specialists. Furthermore, aside from electricity, it should not rely on any other automobile components (such as the odometer or gyros) [33]. This is due to the fact that some vehicles lack interfaces or that older vehicles lack digital interfaces [33].

VII. Conclusion

This paper provides a comparison of Visual SLAM and LiDAR SLAM. We compared modern Visual SLAM techniques such as ORB-SLAM, PTAM, RGB-D SLAM, and Lidar SLAM methods like Scan matching, RBPF, and Graph optimization. Overall, ORB-SLAM algorithm outperformed feature-based MonoSLAM, BA-based PTAM, LSD-SLAM, and other Scan matching and particle filter-based LIDAR SLAM methods. Although RGB-D cameras are popular, they do not work well outdoors, where the ambient light significantly affects detection. Graph optimization LiDAR-based SLAM also yields excellent results in terms of localization accuracy. In general, LiDAR methods appear to produce highly accurate 3D data of the environment, but they are time-consuming and continue to rely on inefficient scan-matching methods. Additionally, LiDAR sensors have a more expensive infrastructure costs and associated hardware costs than cameras. The aforementioned research clearly shows that both LiDAR and Visual SLAM have certain benefits and drawbacks. LiDAR SLAM can operate in low-light or textureless settings while vSLAM cannot. However, LiDAR-SLAM struggles in harsh or rainy environments and textured but geometrically insignificant regions, where camera-based SLAM performs well. We suggest that the combination of LiDAR and Visual SLAM capabilities would be advantageous to the SLAM community. This would make SLAM more robust and resistant to extreme weather and light conditions and enable it to solve a multi-modal, hybrid MAP problem. This literature has not yet examined a LiDAR-vision sensor fusion algorithm, which has to be researched. While some studies have been conducted to fuse vision and LiDAR sensors, they all continue to operate at a very limited fusion level and underutilize the capabilities of both methods. Future work could be done to provide a robust, affordable, tightly hybridized SLAM implementation. Furthermore, we anticipate that such a solution will become more affordable and accurate over time as LiDAR prices have declined over the years and modern visual slam algorithms are becoming more accurate due to sparse bundle adjustment methods.

References

[1] Hugh Durrant-Whyte, and Tim Bailey, “Simultaneous Localization and Mapping: Part I,” Australian Centre for Field Robotics (ACFR) J04, The University of Sydney, Sydney NSW 2006, Australia., IEEE Robotics & Automation Magazine, pp. 99-108, June 2006.

[2] César Debeunne, and Damien Vivet, “A Review of Visual-LiDAR Fusion based Simultaneous Localization and Mapping,” Sensors 2020, MDPI, ISAE-SUPAERO, Université de Toulouse, CEDEX 4, France, April 2020.

[3] Hugh Durrant-Whyte, and Tim Bailey, “Simultaneous Localization and Mapping: Part II,” Australian Centre for Field Robotics (ACFR) J04, The University of Sydney, Sydney NSW 2006, Australia., IEEE Robotics & Automation Magazine, pp. 108-117, September 2006.

[4] Takafumi Taketomi, Hideaki Uchiyama and Sei Ikeda, “Visual SLAM algorithms: a survey from 2010 to 2016,” Springer Open, IPSJ Transactions on Computer Vision and Applications, 2017.

[5] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, Juan Manuel Rendón-Mancha, “Visual simultaneous localization and mapping: a survey,” Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Morelos, México, Springer Science+Business Media Dordrecht, November 2012.

[6] Young-Sik Shin, Yeong Sang Park and Ayoung Kim, “Direct Visual SLAM Using Sparse Depth for Camera-LiDAR System,” IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, May 2018.

[7] Guolai Jiang, Lei Yin, Shaokun Jin, Chaoran Tian, Xinbo Ma and Yongsheng OuA, “Simultaneous Localization and Mapping (SLAM) Framework for 2.5D Map Building Based on Low-Cost LiDAR and Vision Fusion,” Appl. Sci. 2019, MDPI, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, May 2019.

[8] Misha Urooj Khan, Dr. Syed Azhar Ali Zaidi, Arslan Ishtiaq, Syeda Ume Rubab Bukhari, et al., “A Comparative Survey of LiDAR-SLAM and LiDAR based Sensor Technologies,” IEEE, Mohammad Ali Jinnah University International Conference on Computing (MAJICC), 2021.

[9] Randall smith, Matthew Self, and Peter cheeseman, “Estimating Uncertain Spatial Relationships in Robotics,” IEEE International Conference on Robotics and Automation, SRI International, January 2003.

[10] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard, “Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters,” IEEE TRANSACTIONS ON ROBOTICS, VOL. 23, February 2007.

[11] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse, “MonoSLAM: Real-Time Single Camera SLAM,” IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 29, NO. 6, June 2007.

[12] Andrew J. Davison, “Real-Time Simultaneous Localisation and Mapping with a Single Camera,” Ninth IEEE International Conference on Computer Vision (ICCV’03), Robotics Research Group, Dept. of Engineering Science, University of Oxford, UK, April 2008.

[13] Georg Klein ,and David Murray, “Parallel Tracking and Mapping for Small AR Workspaces,” 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Active Vision Laboratory, Department of Engineering Science, University of Oxford, November 2007.

[14] Ping Li, and Zhongming Ke, “Feature-based SLAM for Dense Mapping,” IEEE International Conference on Advanced Mechatronic Systems, Kusatsu, Shiga, Japan, August 2019

[15] Jinbo Sheng , Shunichi Tano, and Songmin Jia, “Mobile Robot Localization and Map Building Based on Laser Ranging and PTAM,” IEEE International Conference on Mechatronics and Automation, Beijing, China, August 2011.

[16] Richard A. Newcombe and Andrew J. Davison, “Live Dense Reconstruction with a Single Moving Camera,” IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Imperial College London, London, UK, June 2010.

[17] Raul Mur-Artal, and J. M. M. Montiel, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System,” IEEE TRANSACTIONS ON ROBOTICS, VOL. 31, NO. 5, October 2015.

[18] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski, “ORB: an efficient alternative to SIFT or SURF,” IEEE International Conference on Computer Vision, Willow Garage, Menlo Park, California, November 2011.

[19] Raul Mur-Artal and Juan D. Tard´os, “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras,” IEEE TRANSACTIONS ON ROBOTICS, VOL. 33, NO. 5, October 2017.

[20] Richard A. Newcombe, Steven J. Lovegrove and Andrew J. Davison, “DTAM: Dense Tracking and Mapping in Real-Time,” IEEE International Conference on Computer Vision, Imperial College London, UK, November 2011.

[21] Jakob Engel, Jurgen Sturm, and Daniel Cremers, “Semi-Dense Visual Odometry for a Monocular Camera,” IEEE International Conference on Computer Vision, TU Munchen, Germany, December 2013.

[22] Jakob Engel, Thomas Sch¨ops, and Daniel Cremers, “LSD-SLAM: Large-Scale Direct Monocular SLAM,” Springer International Publishing Switzerland, ECCV 2014, Part II, LNCS 8690, pp. 834–849, 2014.

[23] Jason Geng, “Structured-light 3D surface imaging: a tutorial,” IEEE Intelligent Transportation System Society, 11001 Sugarbush Terrace, Rockville, Maryland 20852, USA, March 2011.

[24] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, et al., “KinectFusion: Real-Time Dense Surface Mapping and Tracking,” 10th IEEE International Symposium on Mixed and Augmented Reality, Microsoft Research, Basel, Switzerland, October 2011.

[25] Renato F. Salas-Moreno, Richard A. Newcombe, Hauke Strasdat, Paul H. J. Kelly, and Andrew J. Davison, “SLAM++: Simultaneous Localisation and Mapping at the Level of Objects,” IEEE Conference on Computer Vision and Pattern Recognition, Imperial College London, June 2013.

[26] Wolfram Burgard, Cyrill Stachniss, Giorgio Grisetti, Bastian Steder, et al., “A Comparison of SLAM Algorithms Based on a Graph of Relations,” IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, USA, October 2009.

[27] Heon-Cheol Lee, Seung-Hee Lee, Seung-Hwan Lee, Tae-Seok Lee, et al., “Comparison and Analysis of Scan Matching Techniques for Cooperative-SLAM,” IEEE, 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Incheon, Korea, November 2011.

[28] Andrea Censi, “Scan matching in a probabilistic framework,” IEEE International Conference on Robotics and Automation (ICRA), Orlando, Florida, pp. 2291–2296, May 2006.

[29] Kai Lingemann, Hartmut Surmann, Andreas Nuchter, and Joachim Hertzberg, “Indoor and Outdoor Localization for Fast Mobile Robots,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Fraunhofer Institute for Autonomous Intelligent Systems (AIS), Augustin, Germany, October 2004.

[30] Hauke Strasdat, J. M. M. Montiel and Andrew J. Davison, “Real-time Monocular SLAM: Why Filter?,” IEEE International Conference on Robotics and Automation, May 2010.

[31] Hauke Strasdata, J.M.M. Montielb, Andrew J. Davison, “Visual SLAM: Why Filter?,” Science Direct, Image and Vision Computing, Volume 30, Issue 2, pp. 65-77, February 2012.

[32] Jurgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers, “A Benchmark for the Evaluation of RGB-D SLAM Systems,” IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Algarve, Portugal, October 2012.

[33] Nabil Belbachir, Nadia Saad Noori, and Benyamin Akdemir, “Real-time vehicle localization using on-board visual SLAM for detection and tracking,” IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), October 2019.

[34] Fernando Molano Ortiz, Matteo Sammarco, Luis Henrique M. K. Costa, and Marcin Detyniecki, “Applications and Services Using Vehicular Exteroceptive Sensors: a Survey,” IEEE TRANSACTIONS ON INTELLIGENT VEHICLES, pp. 1-20, June 2022.