http://communications.science.ankara.edu.tr

Commun.Fac.Sci.Univ.Ank.Series A2-A3
Volume 65, Number 1, Pages 01-15 (2023)
DOI: 10.33769/aupse.1195379
ISSN 1303-6009 E-ISSN 2618-6462

Research Article; Received: October 27, 2022; Accepted: November 09, 2022

A COMPREHENSIVE COMPUTATIONAL COST ANALYSIS FOR STATE-OF-THE-ART VISUAL SLAM METHODS FOR AUTONOMOUS MAPPING

Ömer Faruk YANIK[1] and Hakkı Alparslan ILGIN[1]

[1]Department of Electrical and Electronics Engineering,
Ankara University, Ankara, TÜRKİYE

Abstract. It is important to solve the autonomous mapping problem with high accuracy using limited energy resources in an environment without prior knowledge and/or signal. Visual Simultaneous Localization and Mapping (SLAM) deals with the problem of determining the position and orientation of an autonomous vehicle or robot with various on-board sensors, and simultaneously creating a map of environment with low energy consumption. However visual SLAM methods require high processing performance for real-time operations. Also, processing capability of the hardware is limited by the power constraints. Therefore, it is necessary to compare the processing load and power consumption of visual SLAM methods for autonomous vehicles or robots. For visual SLAM methods, although there are different comparison studies, there is no comprehensive computational cost analysis covering different datasets and important parameters including absolute trajectory error, RAM Usage, CPU load, GPU load, with total power consumption. In this paper, ORB-SLAM2, Direct Sparse Odometry (DSO), and DSO with Loop Closure (LDSO), which are state of the art visual SLAM methods, are compared. Besides the performance of these methods, energy consumption and resource usage are evaluated allowing the selection of the appropriate SLAM method.

1. Introduction

Autonomous systems have become more visible in daily life, with many products from driverless cars to cleaning robots, from armed unmanned aerial vehicles to consumer electronics. They carry out specific missions autonomously with limited sources providing convenience and assistance to humans.

Keywords. Localization, mapping, indoor navigation, visual SLAM, computational cost.

ofyanik@ankara.edu.tr; 0000-0001-7832-1945
ilgin@eng.ankara.edu.tr-Corresponding author; 0000-0003-0112-4833.

In environments where previously known or satellite signals can be received, relatively uncomplicated autonomous movement capability such as waypoint identification can be achieved with conventional applications. Position information can be obtained through Global Positioning System (GPS) receiver if satellite signal is available. In addition, various devices and sensors are also used to solve the same problem. Among them, Light Detection and Ranging (LIDAR) is used to determine position of the robot. Inertial Measurement Unit (IMU) is another device with accelerometer and gyroscope that is used widely. Position information can be used in a certain standard for functions such as waypoint identification, lane tracking, estimation of the distance passed, obstacle avoidance and distance adjustment. In these applications, the robot, whose spatial relationship is known through its sensors, estimates its own position by using the relevant reference information. However, in an environment where GPS satellite signals cannot be detected by the robot or in a place for which no prior knowledge is available, these methods, due to the high margin of error, are not sufficient in terms of functions such as obstacle avoidance and determining momentary position in its surroundings. In environments that are previously unknown or where GPS signals cannot be detected, different solutions are needed to calculate the robot's position relative to the environment.

There are many studies that include various methods such as motion estimation, target tracking, waypoint tracking based on GPS and different interpretations based on these methods in order to increase autonomous movement capability. However, in order to determine the location of a robot in environments without prior knowledge, it is necessary to obtain information about the environment first and to calculate the position of the robot while obtaining information about the environment.

1.1. **Definition of SLAM Problem.** If there is no information about the surroundings of wheeled robots, measurement of distances by odometer can be used. However, wheel slipping or spinning errors accumulate and increase the total error in the wheel odometry, as the error in the previous position information will also affect the next position information. Additionally, this does not apply to wheelless vehicles and robots. For this reason, different methods based on laser and visual sensors are used to obtain location information. Among these methods, odometry is based on calculating the current position with respect to previously visited locations to follow the course of the robot. On the contrary, for SLAM methods, position and environment are considered together in processing [1].

SLAM is the method that deals with the computational problem of tracking a robot's position while simultaneously creating or updating a map of a previously unknown environment. Since robot has to do two related tasks at the same time, the SLAM problem is a complex issue that needs to be solved in real-time [2].

Calculation of displacement performed by the robot (odometry) and understanding of a place passed again (loop closure) are the basic components of SLAM [3]. In addition to being formulated in many theoretical ways, SLAM has the opportunity to be applied in many areas from indoor robots to underwater and air systems. The solution to the SLAM problem, which enables robots to be truly autonomous, is seen as one of the most significant achievements of robotics field [4].

As well as there are various methods to achieve autonomous mapping solution, there also exists data utilized to evaluate and compare the performance of those methods. Some of the data are obtained in the controlled environments using different techniques. For evaluation in different conditions, there exist different data with specific properties. Thus, it is possible to perform performance tests for visual SLAM methods in various conditions such as surroundings with moving objects.
In the next subsections, information about the probabilistic definition of SLAM, environmental factors, and related works are given.

1.2. **Solution of SLAM Problem.** The position of a robot can be calculated relatively easily if the robot has a precise prior knowledge of the environment. On the contrary, a robust model of the environment in which the robot is located can be created if the position of the robot is known perfectly. However, within the scope of SLAM problem, the robot with on-board sensors does not have any prior knowledge about the environment it is located in. Probabilistic methods are used in solving the SLAM problem, since the robot's location and environment's map are created simultaneously and a perfect information cannot be obtained [3]. In SLAM problem (see Fig. 1), if $\boldsymbol{x_k}$ is the robot's position and orientation, and $\boldsymbol{u_k}$ is the control input applied to bring the robot to position $\boldsymbol{x_k}$ at time $\boldsymbol{k-1}$, $\boldsymbol{m_i}$ is the location of each obstacle that makes up the environment map; $\boldsymbol{z_{ik}}$ refers to observation of the robot at time $\boldsymbol{k}$.

Also, $\boldsymbol{X_{0:k} = \{x_0, x_1, \dots, x_k\}}$ are the positions where the robot has visited in the past. $\boldsymbol{U_{1:k} = \{u_1, u_2, \dots, u_k\}}$ are history of control inputs. Robot's all observations on landmarks are $\boldsymbol{Z_{0:k} = \{z_0, z_1, \dots, z_k\}}$. Together with these definitions, solution of the SLAM problem is based on the probabilistic approach given below as conditional probability ($\boldsymbol{P}$), which must be calculated across whole trajectory:

$$\boldsymbol{P(x_k, m \,|Z_{0:k}, U_{1:k}, x_0)} \tag{1}$$

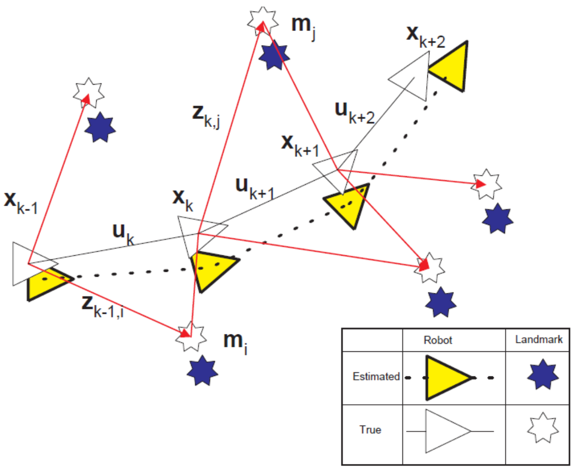

FIGURE 1. Solution of SLAM problem [4].

With the probabilistic definition given in equation (1), many methods have emerged for solution of the SLAM problem. In order to perform these solutions in real-time, extended Kalman filter, particle filter and graph-based methods are mostly used [5].

Besides theoretical solution, autonomous systems in daily life are expected to be able to cope with the usual environmental factors. However, there are many compelling factors within the scope of solving the SLAM problem. Effects such as sensor noise, size of the area where the robot is located, and the dynamic elements in the environment cause mathematical and hardware difficulties in solving SLAM problem. The accuracy of the sensors, as an indicator of the reliability of the data regarding environment and the state of the robot, emerges as an important factor for solution of the SLAM problem. In addition, size of the environment reveals as another problem, as size of the environment will increase the need for memory, processing load, and accumulation of error originating from the sensor. Moving objects, which change model of the environment, are another environmental factor that should be taken into account for solution of the SLAM problem.

In addition to the variety of SLAM's mathematical solution methods, different devices such as camera, accelerometer are used in robots. Robustness of the sensor data will reduce the error accumulated. However, since it is not possible to obtain perfect sensor data in real life, the solution to the SLAM problem should be able to overcome sensor errors and noise. In solution of the SLAM problem, information can be obtained from a single sensor, as well as environmental data can be obtained via multiple sensors of the same type or combinations of different sensors.

1.3. **Visual SLAM.** SLAM methods using camera as sensor are called visual SLAM and are generally divided into two categories, as direct methods that using entire image and the methods utilizing features [6]. To date, many studies have been carried out in single-camera and multi-camera sensor configurations. Among these, as given in Tab. 1, feature-based PTAM [7], ORB-SLAM [8] created by using ORB features developed over [9] PTAM; direct methods, Large Scale Direct Monocular (LSD) SLAM [10], SVO [11], Direct Sparse Odometry (DSO), Direct Sparse Odometry with Loop Closing (LDSO), Dense Visual (DVO) SLAM [12], and ElasticFusion [13] can be counted. There are also many benchmarks used for performance analysis of the visual SLAM problem. These benchmarks are mostly obtained by capturing movement of objects in controlled environments [14] and are separated from each other by several features like outdoor, indoor including moving objects and/or relatively stationary environments. KITTI, EUROC, TUM RGB-D [14], TUM Monocular [15], ICL-NUIM, Sintel [16], Tsukuba [17] and NYU [18] datasets are among them. RAM and CPU usages with processing time on a particular dataset are also compared [19] and for MSCKF [20], OKVIS [21], ROVIO [22], VINS-Mono [23], SVO [11] + MSF [24] and SVO + GTSAM [25]. These studies carried out on four different hardware with Intel and ARM architectures. ATE parameter results are evaluated using some SLAM methods on a wheeled robot with a camera [6]. NVIDIA Jetson TX1 hardware with Ubuntu 16.04 version was used on the robot. LSD SLAM, ORB-SLAM and Direct Sparse Odometry (DSO) were compared. Among the stereo-camera methods, Real-Time Appearance-Based Mapping (RTAB map) [26], ORB SLAM, Stereo Parallel Tracking and Mapping (SPTAM) [27] were compared. ORB2, DynaSLAM [28] and DSO algorithms are also benchmarked [29]. Intel processors were used as part of the analysis. Several parameters including Absolute Trajectory Error were evaluated on TUM Monocular and EUROC data sets.

Although there are many comparison studies, no comprehensive computational cost analysis for visual SLAM methods has been encountered in the literature, apart from comparison of CPU and memory usages for visual-Inertial Navigation methods [19].

TABLE 1. Visual SLAM algorithms.

Algorithm	**Method**	**Map Sparsity**	**Loop Closing**
Parallel Tracking and Mapping (PTAM)	Feature-based	Yes	No
Semi-direct Visual Odometry	Semi-Direct	Yes	No
ORB-SLAM	Feature-based	Semi	Yes
Direct Sparse Odometry (DSO)	Direct	No	No
DSO with Loop Closing	Direct	No	Yes
LSD-SLAM	Direct	No	Yes
DVO-SLAM	Direct	No	Yes
ElasticFusion	Direct	No	Yes

We evaluate performances of state-of-the-art methods such as ORB-SLAM2 (or ORB2) [30], DSO [31], and LDSO [32] on datasets created in different laboratories and separated from each other in terms of information such as indoor and outdoor environments. The most important parameters, which are Central Processing Unit (CPU), Graphics Processing Unit (GPU), and Random Access Memory (RAM) usage as well as power consumption of the SLAM algorithms are compared. On the other hand, related to the performance of visual SLAM methods, this study represents a different and comprehensive perspective using various datasets such as ICL-NUIM, KITTI, EUROC and TUM Monocular, regarding to hardware constraints such as power consumption, memory usage, CPU and GPU load. Material and Methods, Experimental Results, and Conclusion about the study are given in the next three sections.

2. Materials and Methods

In this paper, performance and behaviours of ORB2, DSO, and LDSO algorithms on KITTI, EUROC, TUM Monocular and ICL-NUIM datasets were compared using NVIDIA Jetson TX1 hardware in real-time (online). ATE parameter was obtained via “evo” repository [33] to detect trajectory error and to get graphical results. In order for the datasets to be used by the visual SLAM methods, calibration of the data, definition of the timestep of each image, correct naming of the images, method for reading timesteps are considered. According to SLAM methods, data set formats are also tailored. Analyses were performed on NVIDIA Jetson TX1 hardware which has NVIDIA Maxwell GPU (with 256 NVIDIA CUDA Cores). It has also ARM Cortex Quad Core CPU, 4 GB LPDDR memory with Ubuntu 18.04 operating system. In order to determine the hardware parameters, "tegrastats" software interface provided by NVIDIA JETSON TX1 was used. “tegrastats” interface provides a text file containing detailed information about resources such as RAM usage, GPU and CPU loads and total consumed power in desired period. Tegrastats data obtained at frequency of 1 Hz were converted to .mat file via MATLAB.

2.1. **SLAM Methods Used in Benchmarks.** ORB2 and Direct Sparse Odometry with and without loop closing algorithms are compared in the experimental studies. ORB2, a feature-based SLAM method, can produce solutions with monocular, stereo and RGB-D cameras. On the other hand, LDSO and DSO methods produce monocular solutions. Therefore, monocular solutions were emphasized in the comparisons. ORB2 has map reuse, loop closure and relocalization capabilities. ORB2 can be executed in real-time for synthetic environments as well as indoor and outdoor sequences obtained by cars or robots. It uses ORB features for mapping, tracking, relocalization and loop closing [30].

DSO is a monocular visual odometry method, which can be executed in real-time. DSO combines a full photometric calibration by using lens vignetting, exposure time, and non-linear responses. Without need for features, it is able to sample pixels through the image areas which have density gradient [31].

LDSO method was created using the point selection infrastructure of the DSO method. But it has loop closure feature that enables detection of repeated point. In this respect, LDSO is a monocular Visual SLAM method [32].

2.2. **Benchmark Datasets.** In order to compare SLAM methods, datasets developed in laboratory environments or developed with special equipment are used. There are many datasets available from open sources differentiated from each other according to the environments they describe, and the equipment they use to obtain images. As such, using datasets with different characteristics will be an important approach to reach an accurate result. For this reason, within the scope of this study, KITTI, ICL-NUIM, TUM Monocular and EUROC benchmark datasets, which are available in open sources, were used to compare SLAM methods.

KITTI outdoor data set was captured using various sensors. Therefore, the data set includes images captured by camera beside the measurements obtained via GPS and position information. Also, accelerometer data is supplied [34]. KITTI benchmark provide 11 training and 11 test data, which can also be used with stereo methods (Visual Odometry / SLAM Evaluation, 2012). KITTI data sets named 00, 03 and 07 each with 10 frame per second were used in the comparison of the algorithms ORB2, DSO and LDSO. A micro unmanned aerial vehicle equipped with stereo camera was used to create EUROC data set, which is available publicly. Beside the image data there are also measurement results by accelerator added to the same data set. This data set includes three categories according to degree of difficulty. In the experiments data with various difficulty degrees from this set, which are MH01, V102 and V203 from interior space were used, so that the performance and consumption values of the algorithms are evaluated extensively. For data that is rated from easy to difficult, motion and thus blur increases progressively. ICL-NUIM dataset contains RGB-D data using handheld camera movements [35]. Unlike other datasets, images in the ICL-NUIM dataset were not collected from a real environment. The ICL-NUIM dataset consists of synthetic images and depth information with frequency of 30 Hz, and ground-truth data. ICL-NUIM dataset is divided into two categories as Office Room and Living Room, each with four subsets. It was considered that LivingRoom0 (lR0) and OfficeRoom0 (oR0) which have the maximum number of video frames among the all subsets are appropriate to be used in the analysis. TUM Monocular dataset contains fisheye camera video frames from small indoor to large outdoor environments. It has also ground truth data, and calibration parameters in 50 subsets. Subsets numbered 19, 29, and 30 are used for comparison of LDSO and DSO algorithms. Because ORB2 is not suitable with fisheye camera frame, ORB2 was not compared on TUM Monocular dataset.

3. Experimental Results

In experimental studies, video sequences from TUM Monocular, EUROC, KITTI and ICL-NUIM data sets are used for benchmarking ORB2, LDS and DSO algorithms. Total of five pparameters, which are ATE, RAM usage, CPU and GPU loads and total power consumption are compared in the experiments. In order to avoid non-deterministic results, each SLAM method was executed six times on the video sequences. At the end, average of six results obtained for the parameters was used for the comparisons. Details of the results obtained for the parameters are given in the following subsections.

In order to obtain data that is in-use by RAM, GPU Load, CPU Load, and Consumed Total Power, “Tegrastats” software is used. ATE results are obtained by using Evo software. Evo software supplies ATE parameter as meter (rms). By means of Tegrastats, RAM usage is obtained as Megabyte (MB). GPU Load and Consumed Total Power are determined in term of milliwatt. CPU load is measured as percentage.

3.1. **Absolute Trajectory Error.** ATE is one of the most frequently used parameters in the comparison of SLAM methods. It is an indicator of how well the SLAM method can track ground-truth data. For the calculation of ATE parameter, open-source Evo tool was used, and the trajectory file was compared with the ground-truth data. Results obtained for ICL-NUIM, KITTI, EUROC and TUM Monocular datasets are given in Tab. 2. DSO method gives the best result on OfficeRoom0, while ORB2 gives the best result on all other datasets. Besides, LDSO is more accurate than DSO for TUM Monocular dataset. Trajectory graph of OfficeRoom0 is given in Fig. 2, in which all methods track the ground-truth similar to each other with tolerable differences.

3.2. **RAM Usage.** RAM usage is also an important parameter, which is used to quantify performance of the algorithms. It is aimed to examine RAM load in order to evaluate how SLAM methods use hardware. Therefore, RAM loads created by ORB2, LDSO DSO methods over the video sequences of EUROC, ICL-NUIM, KITTI, and TUM data sets were recorded with the frequency of 1 Hz. RAM usage results are given in Tab. 3. DSO gives the best results in RAM usage due to the lack of loop closure increasing complexity.

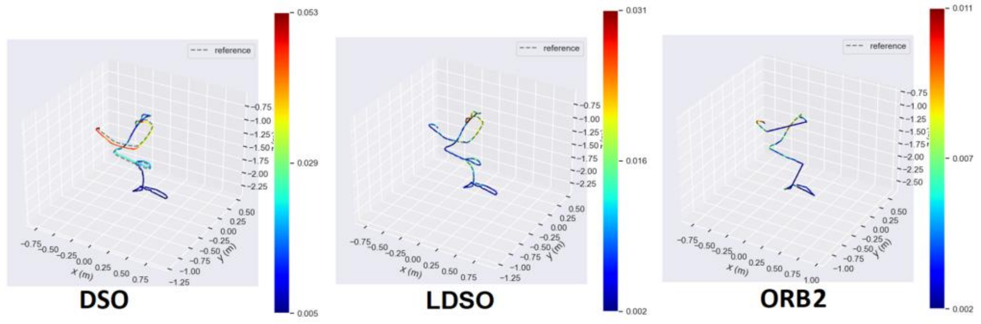

FIGURE 2. Trajectory of methods on officeroom0: DSO (left) gives the best result in terms of ATE.

TABLE 2. ATE results (rms).

	ICL-NUIM		EUROC			KITTI			TUM Monocular		
	oR0	**lR0**	**MH01**	**V102**	**V203**	**00**	**03**	**07**	**19**	**29**	**30**
DSO	0.162	0.024	0.0464	1.026	1.373	119.77	2.314	15.272	0.165	0.147	0.365
LDSO	0.413	0.114	0.0425	1.511	1.142	10.612	2.904	6.385	0.137	0.057	0.085
ORB2	0.171	0.008	0.0441	0.064	0.263	8.3245	1.807	2.132	-	-	-

TABLE 3. RAM usage - MB (rms).

	ICL-NUIM		EUROC			KITTI			TUM Monocular		
	oR0	**lR0**	**MH01**	**V102**	**V203**	**00**	**03**	**07**	**19**	**29**	**30**
DSO	1362	1354	1430	1423	1443	1701	1626	1509	1804	1768	1554
LDSO	1974	1980	2177	2121	2143	2943	2112	2071	2429	2402	2019
ORB2	1515	1547	1558	1525	1524	2160	1457	1475	-	-	-

TABLE 4. CPU usage (rms).

	ICL-NUIM		EUROC			KITTI			TUM Monocular		
	oR0	**lR0**	**MH01**	**V102**	**V203**	**00**	**03**	**07**	**19**	**29**	**30**
DSO	47.292	45.876	45.815	50.562	54.692	61.974	52.661	56.371	57.365	55.167	53.272
LDSO	51.374	49.200	53.796	54.595	55.334	68.162	56.502	59.068	59.114	57.745	57.646
ORB2	55.753	53.813	53.858	52.519	49.349	39.928	46.925	48.067	-	-	-

TABLE 5. Total power consumption – milliwatts (rms).

	ICL-NUIM		EUROC			KITTI			TUM Monocular		
	oR0	**lR0**	**MH01**	**V102**	**V203**	**00**	**03**	**07**	**19**	**29**	**30**
DSO	5558.8	5368.4	6515.6	6250.9	6727.4	8878.6	6070.0	6547.6	8033.2	7377.7	6382.8
LDSO	5747.6	5541.9	6362.5	5940.1	5935.0	6875.9	5926.4	6323.5	6727.0	6297.7	6185.9
ORB2	5351.8	5236.2	5465.1	5248.5	5201.9	4511.5	4959.4	5022.4	-	-	-

3.3. **CPU Usage.** One of the parameters frequently used in the computational load calculations of SLAM methods is CPU load. Since NVIDIA Jetson TX1 has quad-core processor, comparisons were made by averaging the usage rates of four processors. CPU load results are given in Table . LDSO method causes higher CPU usage than DSO for all datasets. This result is about loop closure effect. On the other hand, while ORB2, as a feature-based method, causes more CPU load in case of non-textured environment such as officeRoom0, livingRoom0, and MH01, it causes less CPU usage in more textured environment such as V203, 00, and 07.

3.4. **Total Power Consumption.** One of the most important constraints for autonomous systems is limited power supplies. For this reason, it will be a correct approach to examine how the total power consumption of the hardware changes depending on the SLAM methods, as well as processing load on the CPU. Total power consumption results are given in Table . 5. According to results shown in Table 5, the least power consuming method is ORB2. Furthermore, when compared to DSO, it is obviously seen that LDSO method consumes less power, with an exception of ICL-NUIM dataset on which total power consumption of the methods are similar. This situation is also a question of this study to answer because power consumption behavior is different from the CPU load behavior.

3.5. **GPU Power Consumption.** The difference between CPU usage and overall power consumed behaviors by the hardware, which is created by SLAM methods has revealed the need to check GPU load. The results of given data provided by hardware have been able to explain the differences between CPU loads and total power dissipated.

GPU load in milliwatts of all methods are given in Tab6. For all datasets, GPU power consumption of DSO is higher than the others while ORB2 is the most effective one among all methods for all datasets. GPU power consumption versus time graphs of all methods are also given in Fig. 3. As seen in this figure, GPU load of DSO method for all dataset is getting higher over time. As explained before, on TUM Monocular Dataset, ORB2 methods was not executed**.**

TABLE 6. Power consumption by GPU – milliwatts (rms).

	ICL-NUIM		**EUROC**			**KITTI**		
	oR0	**lR0**	**MH01**	**V102**	**V203**	**00**	**03**	**07**
DSO	346,7439	350,9953	1303,57	1052,36	1472,55	3050,55	621,99	1162,24
LDSO	304,4431	341,4259	902,00	576,95	550,76	995,11	520,67	631,24
ORB2	165,4507	148,8809	99,02	120,71	144,56	81,85	110,61	85,52

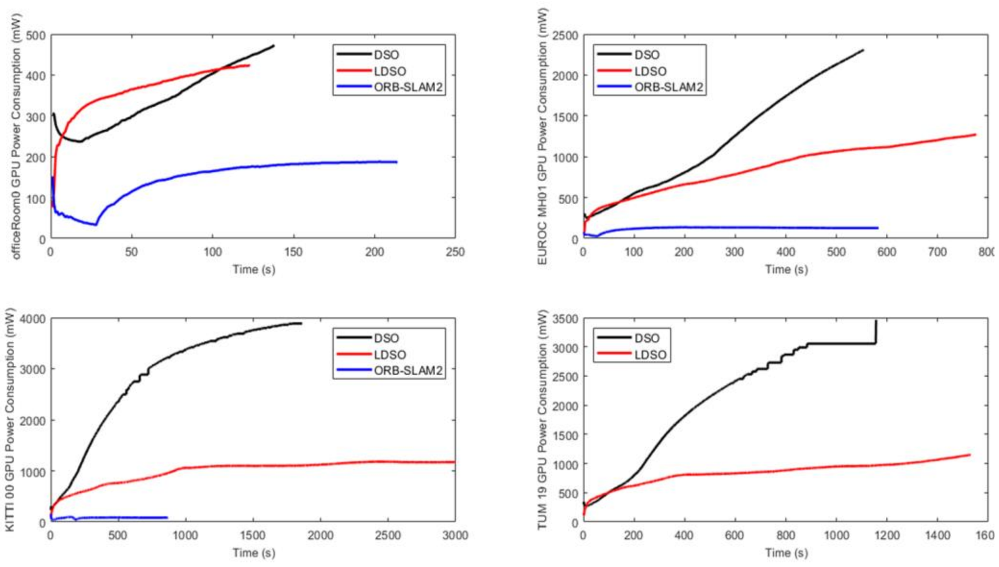

FIGURE 3. GPU power consumption of DSO, LDSO and ORB-SLAM2.

4. Conclusion

In this paper, performances of ORB2, DSO and LDSO methods, which are suitable to be executed on ARM processor that NVIDIA Jetson TX1 has, are compared using ICL-NUIM, KITTI, TUM Monocular and EUROC benchmark datasets. By making an exhaustive comparison, in particular, it is aimed to make a choice among the methods in order to meet the power constraints of mobile robots, localization requirements within indoor environment, and to guide to future studies. ATE, RAM usage, CPU and GPU loads and total consumed power parameters were used for the comparison studies.

ORB2 is the most effective method for ATE parameter, except for the Officeroom0 dataset, for which DSO method gives the best result. The reason for ORB2 gives the worst result for OfficeRoom0 is that it cannot detect features in the Officeroom0 dataset sufficiently.

Due to the lack of loop closure in DSO, it gives the best results in RAM usage as expected. On the other hand, as a direct method, owing to loop-closure ability, LDSO is the method that pushes RAM constraints the most. It has been determined that the LDSO method not only forces the RAM constraint of NVIDIA Jetson TX1 hardware during the process, but also tends to use the SWAP memory more.

As for the CPU load, DSO method gives better results on the ICL-NUIM dataset, while the ORB2 method has better results on the KITTI dataset. However, while DSO method is better for MH01 and V102 dataset, ORB2 method achieves better results on V203 dataset. In this way, it can be said that ORB2 causes more CPU load when solving SLAM problem in surroundings such as supplied by ICL-NUIM where features are less obvious, whereas ORB2 algorithm also causes CPU to have less load in environments where features are dense and can be followed.

When the CPU load and total power consumption are compared, different behaviours are noticed. For instance, CPU is used at least by DSO algorithm on MH01 data set. However, power consumed by this method is the most demanding among all. As an explanation for that, DSO is the method that requires the most GPU power.

For GPU power consumption, the best results were obtained with ORB2 on all datasets while the worst results were obtained by DSO. Differences between GPU power consumption behaviours are related with keyframe frequencies of the methods. DSO and LDSO use 5-10 frames per second [31], and 5-7 frames per second [32], respectively. However, ORB2 uses one frame as a keyframe per 20 frames maximum [30]. This situation is the root cause for the difference between the total power consumption and CPU load.

LDSO method is not suitable to be used in large environments, since it pushed RAM constraints. In textured environments, regardless of the size of the environment, ORB2 is more suitable than DSO and LDSO to solve the SLAM problem within the constraints of the hardware in terms of ATE, GPU and CPU loads, and total power consumption. On the other hand, within the untextured environment with short and straight trajectory, DSO is suitable for the solution of the SLAM problem.

Author Contribution Statements The authors equally worked on the study. All authors read and approved the final copy of the manuscript.

Declaration of Competing Interests The authors declare that there is no conflict of interest regarding the publication of manuscript.

References

[1] Yousif, K., Bab-Hadiashar, A., Hoseinnezhad, R., An overview to visual odometry and visual SLAM: Applications to mobile robotics, *Intell. Ind. Syst.*, 1 (4) (2015), 289-311, https://doi.org/10.1007/s40903-015-0032-7.

[2] Huletski, A., Kartashov, D., Krinkin, K., Evaluation of the modern visual SLAM methods, *Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT),* (2015), 19-25, https://doi.org/10.1109/AINL-ISMW-FRUCT.2015.7382963.

[3] Guçlu, O., Simultaneous Localization and Mapping Using RGB-D Sensors, (2018). Doctoral Thesis, Hacettepe University, Turkey.

[4] Durrant-Whyte, H., Bailey, T., Simultaneous localization and mapping (SLAM): Part I the essential algorithms, IEEE *Robot. Autom. Mag.,* 13 (2) (2006), 99-110, https://doi.org/10.1109/MRA.2006.1638022.

[5] Chong, T. J., Tang, X. J., Leng, C. H., et al., Sensor technologies and simultaneous localization and mapping (SLAM), *Procedia Compt. Sci.,* 76 (2015), 174-179, https://doi.org/10.1016/j.procs.2015.12.336.

[6] Filipenko, M., Afanasyev, I., Comparison of various SLAM systems for mobile robot in an indoor environment, *International Conference on Intelligent Systems (IS),* (2018), 400-407, https://doi.org/10.1109/IS.2018.8710464.

[7] Klein, G., Murray, D., Parallel tracking and mapping for small AR workspaces, *6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, (2007), 225-234.

[8] Mur-Artal, R., Montiel, J. M. M., Tardos, J. D., ORB-SLAM: A versatile and accurate monocular SLAM system, *IEEE Trans. Robot.,* 31 (5) (2015), 1147-1163, https://doi.org/10.1109/TRO.2015.2463671.

[9] Liang, X., Chen, H., Li, Y., Liu, Y., Visual laser-SLAM in large-scale indoor environments, *International Conference on Robotics and Biomimetics*, (2016), 19-24.

[10] Engel, J., Schops, T., Cremers, D., LSD-SLAM: Large-scale direct monocular SLAM,

European Conference on Computer Vision, (2014), 834-849, https://doi.org/10.1007/978-3-319-10605-2_54.

[11] Forster, C., Pizzoli, M., Scaramuzza. D., SVO: Fast semi-direct monocular visual odometry, *IEEE International Conference on Robotics and Automation (ICRA)*, (2014), 15-22, https://doi.org/10.1109/ICRA.2014.6906584.

[12] Kerl, C., Sturm, J., Cremers, D., Dense visual SLAM for RGB-D cameras, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (2013), 2100-2106, https://doi.org/10.1109/IROS.2013.6696650.

[13] Whelan, T., Salas-Moreno, R., Glocker, B., ElasticFusion: Real-time dense SLAM and light source estimation, *Int. J. Robot. Res.*, 35 (14) (2016), 1697-1716, https://doi.org/10.1177/0278364916669237.

[14] Sturm, J., Engelhard, N., Endres, F., A benchmark for the evaluation of RGB-D SLAM systems, *International Conference on Intelligent Robots and Systems*, (2012), 573-580, https://doi.org/0.1109/IROS.2012.6385773.

[15] Engel, J., Usenko, V., Cremers, D., A photometrically calibrated benchmark for monocular visual odometry, (2016), arXiv:1607.02555.

[16] Butler, D. J., Wulff, J., Stanley, G. B., A naturalistic open source movie for optical flow evaluation, *European Conference on Computer Vision*, (2012), 611-625, https://doi.org/10.1007/978-3-642-33783-3_44.

[17] Peris, M., Martull, S., Maki, A., Towards a simulation driven stereo vision system, *Proceedings of the 21st International Conference on Pattern Recognition*, (2012), 1038-1042.

[18] Silberman, N., Hoiem, D., Kohli, P., Indoor segmentation and support inference from RGBD images, *European Conference on Computer Vision*, (2012), 746-760.

[19] Delmerico, J., Scaramuzza, D., A benchmark comparison of monocular visual-inertial odometry algorithms for flying, *International Conference on Robotics and Automation (ICRA)*, (2018), 2502-2509, https://doi.org/10.1109/ICRA.2018.8460664.

[20] Mourikis, A. I., Roumeliotis, S. I., A multi-state constraint Kalman filter for vision-aided inertial navigation, *Proceedings IEEE International Conference on Robotics and Automation*, (2007), 3565-3572, https://doi.org/10.1109/ROBOT.2007.364024.

[21] Leutenegger, S., Lynen, S., Bosse, M., Keyframe-based visual–inertial odometry using nonlinear optimization, *Int. J. Robot. Res.* 34 (3) (2015), 314-334, https://doi.org/10.1177/0278364914554813.

[22] Bloesch, M., Omari, S., Hutter, M., Robust visual inertial odometry using a direct EKF-based approach, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (2015), 298-304, https://doi.org/10.1109/IROS.2015.7353389.

[23] Qin, T., Li, P., Shen, S., Vins-mono: A robust and versatile monocular visual-inertial state estimator, *IEEE Transactions on Robotics*, 34 (4) (2018), 1004-1020,

https://doi.org/10.1109/TRO.2018.2853729.

[24] Lynen, S., Achtelik, M. W., Weiss, S., A robust and modular multi-sensor fusion approach applied to MAV navigation, *IEEE/RSJ International Conference on Intelligent Robots and Systems,* (2013), 3923-3929, https://doi.org/10.1109/IROS.2013.6696917.

[25] Forster, C., Carlone, L., Dellaert, F., On-manifold preintegration for real-time visual-inertial odometry, *IEEE Trans. Robot.,* 33 (1) (2016), 1-21, https://doi.org/10.1109/TRO.2016.2597321.

[26] Labbe, M., Michaud, F., Online global loop closure detection for large-scale multi-session graph-based SLAM, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (2014), 2661-2666, https://doi.org/10.1109/IROS.2014.6942926.

[27] Pire, T., Fischer, T., Castro, G., S-PTAM: Stereo parallel tracking and mapping. *Robotics and Autonomous Systems*, 93 (2017), 27-42, https://doi.org/10.1016/j.robot.2017.03.019.

[28] Bescos, B., Fácil, J. M., Civera, J., DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes, *IEEE Robot. Aut. Let.*, 3 (4) (2018), 4076-4083, https://doi.org/10.1109/LRA.2018.2860039.

[29] Mingachev, E., Lavrenov, R., Tsoy, T., Comparison of ROS-based monocular visual SLAM methods: DSO, LDSO, ORB-SLAM2 and DynaSLAM, *International Conference on Interactive Collaborative Robotics,* (2020), 222-233, https://doi.org/10.1007/978-3-030-60337-3_22.

[30] Mur-Artal, R., Tardós, J. D., ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras, *IEEE Trans. Robot.,* 33 (5) (2017), 1255-1262, https://doi.org/10.1109/TRO.2017.2705103.

[31] Engel, J., Koltun, V., Cremers, D. Direct sparse odometry, *IEEE Trans. Pattern Anal. Mach. Intell.,* 40 (3) (2018), 611-625, https://doi.org/10.1109/TPAMI.2017.2658577.

[32] Gao, X., Wang, R., Demmel, N., Cremers, D., LDSO: Direct sparse odometry with loop closure, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (2018), 2198-2204.

[33] Grupp, M., evo: Python package for the evaluation of odometry and SLAM, (2021). Available at: https://github.com/MichaelGrupp/evo.

[34] Geiger, A., Lenz, P., Stiller, C., Vision meets robotics: The KITTI dataset, *Int. J. Robot. Res.,* 32 (11) (2013), 1231-1237, https://doi.org/10.1177/0278364913491297.

[35] Handa, A., Whelan. T., Mcdonald, J., A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM, *IEEE International Conference on Robotics and Automation (ICRA)*, (2014), 1524-1531, https://doi.org/10.1109/ICRA.2014.6907054.