Embedded Computing for Environmental Sensing and Mapping in Agricultural Robots: A Power and Performance Analysis

1st Andrew M. Roberts
School of Computing
Newcastle University
Newcastle-Upon-Tyne, UK
a.roberts18@newcastle.ac.uk

2nd Tzer-Nan Lin
School of Computing
Newcastle University
Newcastle-Upon-Tyne, UK
t.lin8@newcastle.ac.uk

3rd Christopher J. Holder
School of Computing
Newcastle University
Newcastle-Upon-Tyne, UK
chris.holder@newcastle.ac.uk

4th Ankush Prashar
School of Natural &
Environmental Sciences
Newcastle University
Newcastle-Upon-Tyne, UK
ankush.prashar@newcastle.ac.uk

5th Barnali Das
School of Computer Science & Engineering
University of Sunderland
Sunderland, UK
barnali.das@sunderland.ac.uk

6th Deepayan Bhowmik
School of Computing
Newcastle University
Newcastle-Upon-Tyne, UK
deepayan.bhowmik@newcastle.ac.uk

***Abstract*—Recent trends in agriculture show decreasing availability of agricultural workers, coupled with reduction in productivity and yield due to climate change. Autonomous agricultural robots are a suitable technology to counter these issues, but in order to gain widespread use they must be able to effectively localise themselves in complex rural environments, whilst being able to run for long periods of time. This paper aims to aid in the selection of Simultaneous Localisation And Mapping (SLAM) variants and parameters when used in the agricultural field in order to maximise range/operational time with minimal sacrifice of effectiveness as evaluated by Percentage Tracked Path (PTP) and Average Trajectory Error (ATE). We investigate some of the most promising applications of ORB-SLAM algorithms, running on a low-resource system, providing insights into the energy costs of increased real-time performance in terms of Frames Per Second (FPS) achieved through varying SLAM parameters. Our contributions are: a full Energy Per Frame (EPF) analysis over all ORB-SLAM parameters; SLAM analysis of a benchmark LFSD dataset, entirely performed on a common low-resource robotics computer, Jetson Nano; evaluation of the determined parameters to achieve highest PTP at lowest EPF; and live trials on the Turtlebot robotic platform using the determined parameters. This work demonstrates that the most power efficient and accurate performance of the variants tested can be attained using ORB-SLAM3 with the number of features reduced to 600. This improved on default parameter operation by 20% on EPF and 8.21% on PTP. This outcome was corroborated in live testing achieving a similar 14.4% EPF drop to $321.83 \pm 46.14\,\mathrm{mJ}$. Therefore, utilising our power performance evaluation, power can be optimised for real-time usage based on the parameters selection, enabling a more accurate and power-efficient SLAM for longer operation time in the field.**

***Index Terms*—Embedded Computing, SLAM Optimisation, Low-Resource Robotics, Power-Analysis, Agricultural Robotics**

I. Introduction

The field of autonomous robotics has advanced dramatically in recent years, though agricultural environments still present a challenge due to their unstructured nature, uneven terrain, changeable weather and dynamic lighting conditions, which can cause issues with machine vision perception [1] and motion control. To this end, smarter systems are required to navigate autonomously in the agricultural environment with Artificial Intelligence (AI) techniques being a solution to this. However, these typically rely on high computing power performed onboard high-end computers or have computation performed off-system via internet connection. As many farms have limited/no internet connection, this is a problem preventing widespread adoption. Therefore, an aim of robotics within the agricultural sector is to be able to operate autonomously with all processing performed on-board, bringing with it the problem of power limitations. Power ultimately affects the rate at which data can be analysed and acted on, determining the speed, range and operational time of the robot.

Fig. 1: Example image from LFSD, presented in grayscale.

The most popular solution to robot navigation is Simultaneous Localization and Mapping (SLAM [2]). SLAM technology endows robots with the capability to self-localize and construct environmental maps in unknown or dynamically changing environments. SLAM enables robots to achieve genuine autonomous navigation even in locations lacking external navigation signals. However, the implementations of SLAM typically require high-end computers with high energy cost, making them unsuitable for applications in low-resource robotics in agriculture. While previous works [3] have evaluated the energy efficiency of SLAM algorithms, they have not explored how

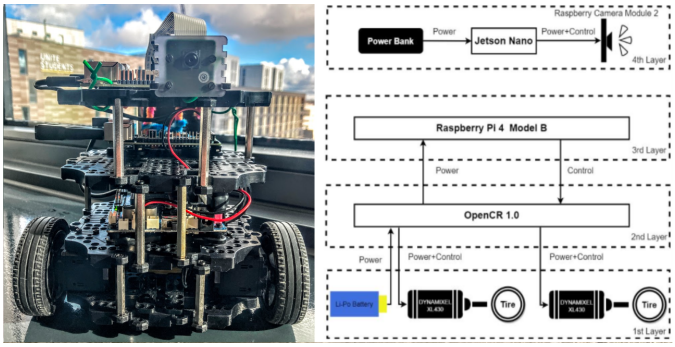

Fig. 2: Left: Our TurtleBot platform. Right: Hardware breakdown of the system.

varying the parameters of the SLAM algorithm impact its energy performance on a particular device, in conjunction with its accuracy.

Our contributions are: an Energy Per Frame (EPF) analysis over three different ORB-SLAM variations and all five ORB-SLAM parameters individually, entirely performed on a common low-resource robotics computer (Jetson Nano [4]) with the LFSD [5] dataset; optimisation analysis of the determined most-effective parameters to achieve highest PTP at lowest EPF; and a live robotics trial of the most power efficient configuration on an indoor environment. Thereby enabling improved operation time in the field through parameter tuning.

II. Background

In this work we aim to improve SLAM utilisation in the agricultural environment, to unlock greater range and operation time for low-resource robotics. This is crucial to bring the cost of autonomous agricultural robots down, allowing widespread adoption to alleviate labour shortages in the agricultural sector and assist in dealing with increasing environmental and financial pressures.

Our test platform, TurtleBot Burger[1], is an open-source robotic platform which allows researchers and scholars to access and modify its design and software, fostering advancements in robotics. Its modularity facilitates the flexible addition or replacement of hardware components and guarantees software scalability. The foundational configuration of TurtleBot encompasses OpenCR 1.0, Dynamixel, LIDAR, and IMU. ROBOTIS developed OpenCR 1.0 to work seamlessly with Robot Operating System (ROS)[2], simplifying robot development and testing, which we use for teleoperation. Fig. 2 shows our initial setup with the Turtlebot, using a Jetson Nano mounted on the top platform with an attached external power supply, allowing us to measure power only related to SLAM without interference of running motors and other peripherals.

SLAM primarily aims to place a mobile robot in an unknown environment at an unknown location. The robot then gradually constructs a consistent map of this environment while determining its position within that map [6]. ORB-SLAM [7], Oriented FAST and Rotated BRIEF-SLAM, is a high-performance system suitable for monocular, stereo, and RGB-D. There are three main stages during ORB-SLAM operation: tracking, local mapping, and re-localisation. ORB-SLAM estimates the camera movement between the current and previous frames during the tracking stage. The local mapping phase optimizes the structure and motion of observed keyframes and three-dimensional landmarks to ensure accuracy and robustness. The re-localisation stage aims to correct the camera's deviation from its original path and guides it back to a previously recognized position.

There are currently three iterations of ORB-SLAM: the first-generation ORB-SLAM, ORB-SLAM2 [8], and ORB-SLAM3 [9]. ORB-SLAM was omitted due to ORB-SLAM2 having undergone many optimizations, and whilst ORB-SLAM3 also introduces enhanced features, both ORB-SLAM2 and ORB-SLAM3 have been proven in research to offer exceptional performance and robustness [10]. Therefore, ORB-SLAM2 and ORB-SLAM3 sufficiently meet the requirements of the current study. In addition, there are multiple CUDA enhanced versions of ORB-SLAM2 available. We used a publicly available version[3] from github licenced under GNU GENERAL PUBLIC LICENSE.

III. Methodology

We hypothesise that power usage of the SLAM algorithms will vary according to the initial parameters selected, and that by recording the power usage along with key metrics for reliability and accuracy, optimised states may be found enabling low power operation without significantly sacrificing accuracy. To evaluate this hypothesis we developed experiments comprising; firstly, the SLAM algorithms running on a suitable agricultural dataset in order to determine the optimal parameters; secondly, running the combined optimal parameters to determine their combined effect; and thirdly, a live run on a robotic platform to showcase the transferability and relevance of this approach to real-world applications.

Hardware: The three ORB-SLAM algorithms have been installed on a Jetson Nano P3450 running Jetpack 4.6.4, with a Raspberry Pi Camera Module 2 for live usage. The Jetson Nano is recommended to be used with NVIDIA's Jetpack OS, which is specialised for the device and comes supplied with many packages for AI usage and CUDA-enhancement. Jetpack 4.6.4 was used as it is a proven stable version. The different ORB-SLAM versions were installed in parallel and able to run without conflict. The Jetson nano has multiple power supply options; when supplied via a USB micro-C cable, the system can be run in a 5W mode and a MAXN(10W) mode, however, the 10W mode is throttled. To unlock the full power potential of the Nano, the barrel jack is connected with jumper J48 in place, which allows MAXN mode to be run without throttling. This is the setup we used for dataset experimentation, whilst for live experimentation, a 5V 4.5A battery pack with USB connection in MAXN mode with throttling was used.

Power measurement: The method of power measurement used here is based on the inbuilt INA3221 power monitor. This sensor writes power measurements to files in the Linux system, which can be polled during testing. This method proved to be a quick and automated method to record power usage throughout the experiments. An initial power baseline was recorded whereby the Jetson Nano was left idle for one minute and power was recorded at 10Hz, resulting in 2741.69 ± 173.65 mW for dataset configuration and 2221.33 ± 384.84 mW for live configuration. All power readings reported in Section IV are during SLAM operation (initialisation phase excluded) and have the baseline subtracted in order to represent only the extra power used by running the SLAM algorithms.

Metrics: The main focus of this paper is power usage, put into the context of effect on FPS. A value of Energy Per Frame (EPF):

$$EPF[\text{mJ}] = \frac{\text{Power [mW]}}{\text{Frames Per Second } [\text{s}^{-1}]} \quad (1)$$

[1]https://www.turtlebot.com/turtlebot3/

[2]https://ros.org/

[3]https://github.com/thien94/ORB_SLAM2_CUDA

is used. This allows the roboticist to put their choice of algorithm and parameter values into the context of the trade-off between power and FPS. Reliability is key, and with parameters being explored that may result in lost tracking, it is also necessary to present a value that represents the reliability of tracking called the Percentage Tracked Path (PTP):

$$PTP[\%] = \frac{\text{Total SLAM Tracked Time [s]}}{\text{Ground Truth Time [s]}}. \quad (2)$$

The accuracy during these runs is also important, and for this the Average Trajectory Error (ATE) is calculated using the evo[4] Python package with the inbuilt Umeyama method which corrects for monocular scale invariances.

Algorithmic optimisation/adaption for embedded use: Here we investigate if there is a critical value for each parameter that allows a lower EPF without sacrificing accuracy. The hypothesis is that by performing a full parameter sweep with the desired dataset environment, a more power efficient operating mode can be attained. The different parameters for ORB-SLAM are described below:

- **Features:** controls the maximum number of keypoints extracted per frame, impacting feature matching and tracking robustness. A higher number increases accuracy but also computational cost, affecting real-time performance.
- **IniThFAST:** initial threshold for the FAST feature detector, determines the sensitivity of keypoint detection. Lower values detect more features in low-texture regions, while higher values prioritise stronger, more distinct keypoints.
- **MinThFAST:** minimum threshold for FAST detection, used when the initial threshold fails to find enough features. Lowering this value ensures keypoint detection in challenging environments, but may introduce weaker features.
- **Pyramid Scale Levels:** defines the number of image scale levels used in the feature detection pyramid, affecting multi-scale feature tracking. More levels improve robustness to scale variations but increase processing time.
- **Scale Factor:** determines the downsampling ratio between consecutive pyramid levels. A lower value results in finer resolution changes, while a higher value speeds up computation but may lose small-scale details.

Highest PTP parameters running on LFSD dataset: For each algorithm the parameters with the highest PTP values, or those with lower EPF for similar PTP, will be combined and used to run 10 repetitions, comparing the results to the default values.

Best parameters running live in indoor environment: A live indoor trial is designed to run using the most efficient SLAM version on the Turtlebot in order to prove the relevance and effectiveness of this approach. The Jetson Nano was mounted on the Turtlebot as shown in Fig. 2 with power supplied via power bank. The Turtlebot was teleoperated down a straight path in an office space, analysing the power usage only during the fully tracked sections of the run.

IV. Experimental Results and Discussions

Dataset Selection: An agricultural dataset was preferred over more common benchmarking datasets like TUM, due to the nature of the different features present in agricultural environments. The Rosario Dataset is a popular, publicly available dataset for this environment, but the images proved to be too low quality for monocular slam to be able to initialise tracking, similar results were found here [5]. Therefore, a less common dataset with higher quality images was found in the Lettuce For SLAM Dataset [11]. Though the camera angle is not optimal for robot navigation, the images e.g. Fig. 1 show early-growth stage crop row environment from the point of view of a row-following robot. The ground truth FPS was only 7.5, so the timestamps were modified before testing in order to allow SLAM to run at up to 75 FPS. We use the dataset in monocular, grayscale mode.

[4]github.com/MichaelGrupp/evo

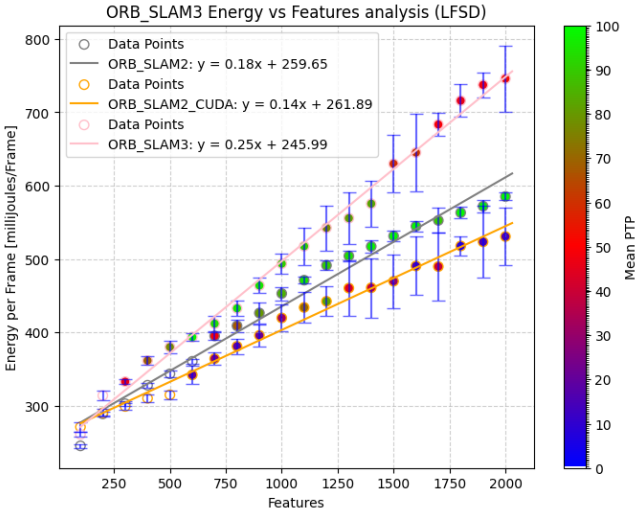

Fig. 3: Number of features effect on EPF.

TABLE I: Best PTP Performance by Features.

ORB-SLAM	Features	PTP[%]	EPF[mJ]	ATE[m]
2	2000	99.32	585.18	0.493
2-CUDA	1200	81.92	442.35	0.327
3	600	98.88	393.59	0.434

A. Algorithmic optimisation for embedded computation

In all graphs, the colours of the data points are representative of Percentage Tracked Path (PTP), with the colour map of values presented to the right of the plot.

Number of features: varied between 100-2000 in steps of 100. Default value is 1000. Fig. 3 shows the general trend between all ORB-SLAM variations that a higher number of features results in a higher energy cost per frame, as would be expected. However, it does not always lead to a higher PTP. In fact, only in ORB-SLAM2 does this hold. By using this analysis it was found that for ORB-SLAM2 although 2000 features resulted in the highest PTP (99.32%), a very similar result is obtained at 1200 features resulting in an energy saving of 93.79 ± 8.4mJ for a drop in 3.74% PTP. For ORB-SLAM3 it was found that as number of features increased over 900 the PTP result worsened, even with a higher power usage. Table I shows the feature numbers resulting in the highest PTP and the associated EPF and ATE. The ATE is quite considerable, as would be expected from monocular-based SLAM without any other sensor inputs. Monocular-based SLAM relies only on the movement between frames to estimate depth and because there are no other inputs like an Inertial Measurement Unit to corroborate distance estimations with our setup then these errors accumulate over time. We therefore find a linear relationship between PTP and ATE; longer tracked path resulting in more drift accumulation and hence worse ATE. This is why PTP is presented as the primary metric, with ATE presented for

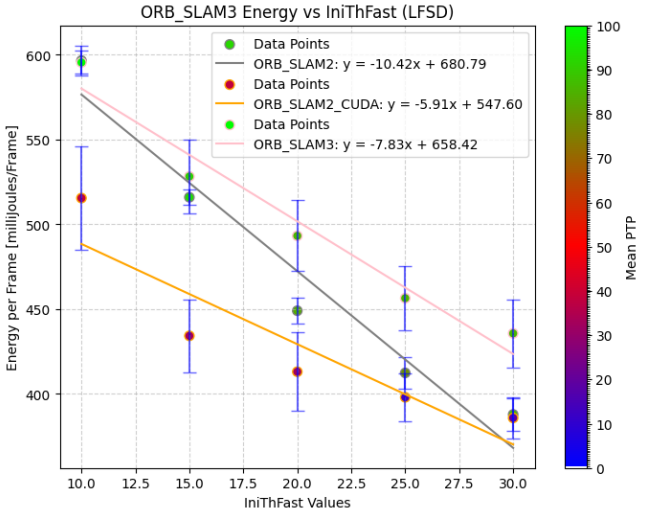

Fig. 4: IniThFAST effect on EPF.

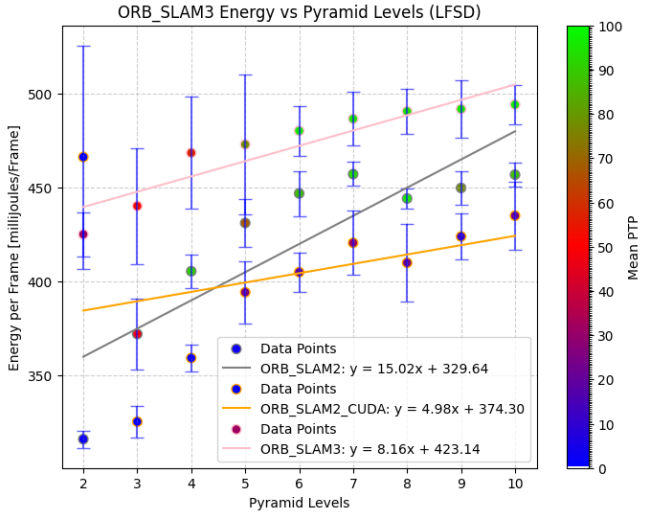

Fig. 5: Pyramid level effect on EPF.

TABLE II: Best PTP Performance by IniThFAST Value.

ORB-SLAM	IniThFAST	PTP[%]	EPF[mJ]	ATE[m]
2	15	93.43	516.01	0.370
2-CUDA	10	37.87	515.53	0.080
3	10	99.82	595.54	0.444

TABLE III: Best PTP Performance by Pyramid Levels.

ORB-SLAM	Levels	PTP[%]	EPF[mJ]	ATE[m]
2	8	96.34	444.02	0.407
2-CUDA	7	31.22	420.36	0.117
3	6	95.37	480.09	0.432

comparison at similar PTP levels. In our results in Table I we see the linear trend overall but also that ATE for ORB-SLAM2 is 14% higher for only 0.4% PTP increase compared to ORB-SLAM3. This would likely indicate that the ORB-SLAM3 produces less drift errors than ORB-SLAM2.

Initial FAST threshold value: IniThFast values were varied between 10-30 in steps of 5. Default value is 20. The general trend is that a higher IniThFast value results in lower EPF. ORB-SLAM2-CUDA is the most power efficient overall but only manages to track less than 50% of the path. ORB-SLAM2 and ORB-SLAM3 manage to track most of the path with ORB-SLAM2 having a $79.53 \pm 8.2\,\mathrm{mJ}$ improvement for a 6.39% drop in PTP, as shown in Table II. The plot also shows that ORB-SLAM2 and ORB-SLAM3 EPF values can be greatly improved with minimal loss in PTP by increasing the IniThFast value to 30.

Minimum FAST threshold value: MinThFast values were varied between 5-20 in 5 unit steps. Default value is 7. No statistical difference was found for these values, likely indicating that with this dataset enough features were found with the IniThFAST value of 20 and so the MinThFAST values were not used.

Pyramid levels: Pyramid Levels were varied between 2-10 in 1 unit steps. Default value is 8. The data shows that decreasing the number of pyramid levels results in lower EPF, as would be expected as there is less computation required. The same general trend of the energy efficiency between the variants holds here as seen in Fig. 5, once again the CUDA-enhanced version shows worst PTP values, with ORB-SLAM2 and ORB-SLAM3 PTP only rising above 80% when the pyramid levels increase past 4 and 6 respectively. Table III shows ORB-SLAM2 is able to beat ORB-SLAM3 for energy efficiency with a higher PTP, but further analysis of the data shows that ORB-SLAM2 achieves a further 74.85 ± 10.22 mJ improvement over ORB-SLAM3 by dropping pyramid levels to 4.0 with only a 4.2% drop in PTP.

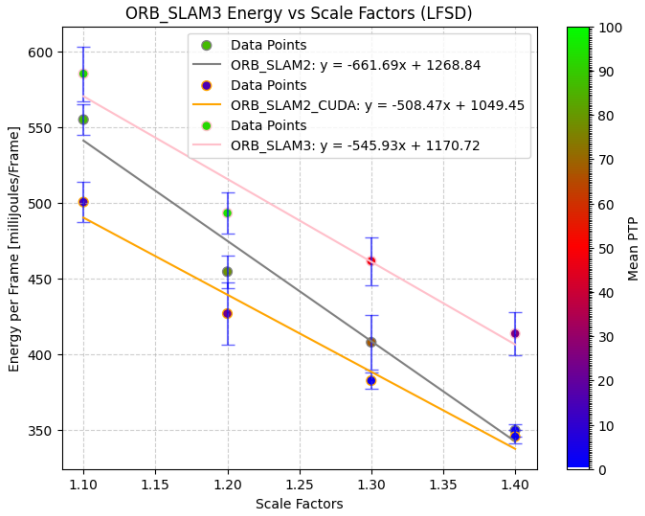

Fig. 6: Scale factor effect on EPF.

TABLE IV: Best PTP Performance by Scale Factors.

ORB-SLAM	Scale Factor	PTP[%]	EPF[mJ]	ATE[m]
2	1.1	86.97	554.80	0.355
2-CUDA	1.2	23.11	426.80	0.083
3	1.2	94.36	493.14	0.363

Scale factors: Scale Factors were varied between 1.1-1.4, in steps of 0.1. Default value is 1.2. A higher scale factor results in lower EPF as there are less scaling operations between the pyramid levels. The intervariant EPF relation remains the same in Fig. 6. ORB-SLAM2 and ORB-SLAM3 PTP only becomes acceptable below a scale factor of 1.2, best PTP values shown in Table IV.

TABLE V: ORB-SLAM Performance Comparison for Default (def) and Combined Optimised (opt) Parameters.

ORB-SLAM Algorithm	EPF Mean [mj]	EPF Error[mj]	ATE Mean[m]	ATE Min[m]	ATE Stdev[m]	PTP Mean[%]	PTP Max[%]
ORB-SLAM2(def)	452.97	8.98	0.345	0.010	0.128	85.37	98.90
ORB-SLAM2(opt)	**491.10**	**12.79**	**0.479**	**0.011**	**0.324**	**70.65**	**100.00**
ORB-SLAM2-CUDA(def)	419.69	18.21	0.127	0.008	0.188	32.44	98.90
ORB-SLAM2-CUDA(opt)	**465.47**	**27.03**	**0.271**	**0.012**	**0.208**	**64.76**	**100.00**
ORB-SLAM3(def)	493.64	13.48	0.355	0.038	0.157	90.69	98.90
ORB-SLAM3(opt)	**491.20**	**10.93**	**0.292**	**0.028**	**0.186**	**91.56**	**100.00**

B. Highest PTP Parameters with LFSD Dataset

Here we combined the parameter values that resulted in the best PTP values, as summarised in Tables I-IV, apart from ORB-SLAM2 Pyramid Levels, where 4.0 was used rather than 8.0 for a $38.78 \pm 10.22\,\mathrm{mJ}$ saving, and features, where 1200 rather than 2000 was used for a $93.79 \pm 8.4\,\mathrm{mJ}$ saving.

From Table V we can see that combining all of the best performing-parameters at once did not yield significantly better results that running the baseline, default parameters. In fact ORB-SLAM2 decreased in PTP and increased in EPF. The PTP Max column shows that whilst the default values never resulted in 100% PTP, the improved parameters did at least sometimes fully track the path. From these results the most appropriate SLAM version is ORB-SLAM3 which has a large PTP advantage over the other variants with almost equal EPF compared to ORB-SLAM2. This is surprising as from all parameter sweeps the ORB-SLAM3 was consistently the highest EPF per value investigated. However, it seems that ORB-SLAM3 is able to undercut ORB-SLAM2 by using more power efficient parameter values to achieve equal PTP results. Also of interest is that for both ORB-SLAM3 and ORB-SLAM2-CUDA, significantly better PTP and EPF values were obtained by decreasing only the number of features.

C. Best Parameters with Live Trial

Using the results from the previous section, it was found that the best combined result of PTP and EPF was achieved by the ORB-SLAM3 algorithm. Therefore this was chosen as the most appropriate configuration for a live test in a Newcastle University office space. The test was run once with the default settings, and again with the features adjusted to 600. Analysing the data post-initialisation until the tracking was lost at the end of the run, the results showed roughly equal power consumption each time but the FPS increased from 14.8 ± 2.0 to 16.9 ± 2.3 due to reducing the features, which reduced the EPF from $376.07 \pm 52.31\,\mathrm{mJ}$ to $321.83 \pm 46.14\,\mathrm{mJ}$, a statistically significant $54.24\,\mathrm{mJ}$/14.4% saving, which is comparable to the $100.05\,\mathrm{mJ}$/20.0% saving determined from dataset analysis alone. This indicates that this approach is able to approximate the energy savings that can be achieved through parameter optimisation. Example videos of the runs in the lab environment are available[5] as well as the complete code and raw data in github[6].

V. Conclusion

This paper has shown the variability of energy costs related to the five parameters of ORB-SLAM2, ORB-SLAM2-CUDA and ORB-SLAM3 and their effect on PTP and ATE. In particular, it has shown that a higher accuracy does not always come at the cost of higher power and slower FPS. In fact, increasing the number of features, and therefore power used, resulted in a worsening of the algorithms reliability in the form of PTP. In the tested LFSD dataset, reducing feature values from 1000 to 600 was sufficient with ORB-SLAM3 to result in the lowest EPF/highest PTP combination, with a 21% improvement on EPF and 7.14% improvement in PTP. It also showed that it was not ultimately necessary to examine all parameters in depth and that the 'Number of Features' parameter was the most effective parameter to optimise for power/accuracy trade-offs. Finally, we showed that the results determined from dataset analysis were comparable to those found from live testing with the Turtlebot platform, with a 14.4% EPF improvement on ORB-SLAM3's default configuration. Our next steps would be to investigate why CUDA-enhancement resulted in worse PTP results compared to the normal CPU variant, as it was consistently the most power-efficient, as well as investigating if this approach has similar results across different datasets. In addition, we would like to utilise heuristic algorithms such as A^* to explore optimisation with not just parameters but also other visual SLAM variants (LSD-SLAM, DROID-SLAM), hardware (Raspberry Pi 5, other Jetson devices) and heterogeneous architectures (CPU+GPU, CPU+FPGA) to improve power efficiency.

[5] 10.25405/data.ncl.29064284

[6] github.com/AndyMRoberts/SAS_2025

References

[1] Yuhao Bai, Baohua Zhang, Naimin Xu, Jun Zhou, Jiayou Shi, and Zhihua Diao, "Vision-based navigation and guidance for agricultural autonomous vehicles and robots: A review," *Computers and Electronics in Agriculture*, vol. 205, pp. 107584, 2023.

[2] H. Taheri and Z. C. Xia, "SLAM; definition and evolution," *Engineering Applications of Artificial Intelligence*, vol. 97, pp. 104032, 2021.

[3] Lu Chen, Gun Li, Weisi Xie, Jie Tan, Yang Li, Junfeng Pu, Lizhu Chen, Decheng Gan, and Weimin Shi, "A survey of computer vision detection, visual slam algorithms, and their applications in energy-efficient autonomous systems," *Energies*, vol. 17, no. 20, 2024.

[4] Gordeev Alexey, Vladimir Klyachin, Kurbanov Eldar, and Aleksandr Driaba, "Autonomous mobile robot with AI based on jetson nano," in *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 1*. Springer, 2021, pp. 190–204.

[5] Fangwen Shu, Paul Lesur, Yaxu Xie, Alain Pagani, and Didier Stricker, "SLAM in the field: An evaluation of monocular mapping and localization on challenging dynamic agricultural environment," 2020.

[6] Hugh Durrant-Whyte and Tim Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[7] Raul Mur-Artal, Juan D. Tardós, and J. M. M. Montiel, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[8] Raul Mur-Artal and Juan D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[9] Carlos Campos, Richard Elvira, Juan J. G. Rodríguez, J. M. Montiel, and Juan D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[10] Dmitry Sharafutdinov, Mikhail Griguletskii, Pavel Kopanev, Mikhail Kurenkov, Gonzalo Ferrer, Andrey Burkov, and Dzmitry Tsetserukou, "Comparison of modern open-source visual SLAM approaches," *Journal of Intelligent & Robotic Systems*, vol. 107, no. 3, pp. 43, 2023.

[11] Shuo Wang, Daobilige Su, Maofeng Li, Yiyu Jiang, Lina Zhang, Hao Yan, Nan Hu, and Yu Tan, "LFSD-dataset," 2023.