Resource-aware Online Parameter Adaptation for Computationally-constrained Visual-Inertial Navigation Systems

Pranay Mathur[1], Nikhil Khedekar[2], and Kostas Alexis[2]

***Abstract*— In this paper, a computational resources-aware parameter adaptation method for visual-inertial navigation systems is proposed with the goal of enabling the improved deployment of such algorithms on computationally constrained systems. Such a capacity can prove critical when employed on ultra-lightweight systems or alongside mission critical computationally expensive processes. To achieve this objective, the algorithm proposes selected changes in the vision front-end and optimization back-end of visual-inertial odometry algorithms, both prior to execution and in real-time based on an online profiling of available resources. The method also utilizes information from the motion dynamics experienced by the system to manipulate parameters online. The general policy is demonstrated on three established algorithms, namely S-MSCKF, VINS-Mono and OKVIS and has been verified experimentally on the EuRoC dataset. The proposed approach achieved comparable performance at a fraction of the original computational cost.**

I. INTRODUCTION

Robotic systems and especially Micro Aerial Vehicles (MAVs) owe their autonomous capabilities largely due to the progress in the domain of localization and mapping methods capable of running online and onboard. Among the multiple sensor fusion strategies employed, fusing visual cameras, LiDAR sensors and more, the combination of vision and inertial sensors is an appealing solution due to its small size, low weight, low cost and demonstrated ability to estimate the robot pose with accuracy and robustness. Major successes in the domain have been demonstrated through important milestones, including the ability to provide reliable odometry in both outdoor [1] and indoor [2] settings, reliability in fast navigation [3–5], resilience against visual degradation [6–8] and more. The community has particularly focused on robust Visual-Inertial Odometry (VIO) solutions, while selected works aim to address the overall Simultaneous Localization And Mapping (SLAM) problem [9, 10]. Despite the progress, however, visual-inertial odometry estimation remains computationally expensive and necessitates significant computational resources onboard the flying robot. This in turn prohibits the potential of integrating such systems onboard ultra lightweight systems or the ability to run alongside other computationally expensive processes (e.g., planning) in computationally-constrained systems.

[1] Pranay Mathur is an undergraduate student pursuing Electronics and Instrumentation Engineering at Birla Institute of Technology and Science (BITS) Pilani Goa Campus, India f20170487@goa.bits-pilani.ac.in

[2] The authors are with the Autonomous Robots Lab, Norwegian University of Science and Technology, O. S. Bragstads Plass 2D, 7034, Trondheim, Norway.

In response to the above, this work contributes a general policy to automatically adapt the behavior and computational profile of a selected set of successful VIO frameworks in order for them to maintain accuracy and robust performance, while self-adjusting their functionality to best fit onboard computationally-constrained systems and respond to real-time changes in the overall available resources as other threads run simultaneously. More specifically, the proposed approach on Computational Resources-aware Visual-Inertial Odometry Scheduling (CRVIOS), is demonstrated in connection to three established VIO methods, namely a) Stereo Multi-State Constraint Kalman Filter (S-MSCKF) VIO [5], b) Robust and Versatile Monocular Visual-Inertial State Estimator (VINS-Mono) [3] and c) Open Keyframe-based Visual-Inertial SLAM (OKVIS) [11].

For all three methods, we define a unifying policy, that given different computing systems and online profiling of available resources, proposes selected changes on the vision front-end and optimization back-ends of these methods both for compile-time variables such as the image resolution, and online real-time manipulation of parameters including the number of features tracked, the number of iterations in the optimization steps and the ratio of images processed. The method further relates its online parameter manipulation to the motion dynamics experienced by the robot at any given time. The proposed VIO scheduling functionality achieves almost identical performance to the original methods but often at a fraction of the computational cost hence enhancing the ability to be deployed onboard extremely lightweight, power-efficient processing boards.

To demonstrate the potential and capabilities of the method, we present a set of experimental case studies. Exploiting the widely-adopted EuRoC dataset [2], we demonstrate CRVIOS for the selected VIO methods on both a) a low-key x86 architecture processor, and b) a low-power ARM system. Specifically, an Intel Core i3-4010U with two cores at 1.7GHz and a Raspberry Pi 4B integrating a Quad core Cortex-A72 (ARM v8) 64-bit SoC at 1.5GHz are considered. For both systems, we run tests by launching additional processes in real-time, thus modifying the actual available resources, and evaluating the resilience of CRVIOS in delivering accurate results by adjusting the parameters of the underlying estimation frameworks.

The remainder of this paper is organized as follows: Section II outlines related work. The proposed approach is detailed in Section III with the implementation details provided in Section IV. Evaluation studies are detailed in Section V, followed by conclusions drawn in Section VI.

II. RELATED WORK

With respect to how different approaches lead to reduced computational cost, most computationally lightweight VIO methods can be broadly classified into two categories. The first relies heavily upon implementation and targets lowering the complexity of the computations or a reduction in time taken by reorganization, pipelining and parallelization [12]. The other relies upon researchers altering a) the front-end by choosing to process only selective information-rich features and landmarks [13, 14] or b) the back-end by graph sparsification [15, 16]. The authors in [17] propose an entropy-driven metric to select only the most informative measurements and achieve a $10\times$ reduction in computation, although the method has been developed for use in AR/VR glasses and well-lit settings. Certain keyframe-based methods discard all other measurements and only process measurements from a subset of spatially distributed camera poses [18, 19]. Adjusting the sensitivity of feature detection while simultaneously altering the back-end to switch between EKF-based SLAM and MSCKF [20] to fully utilize the computational resources was shown in [21]. A similar resource-aware approach is assigning different portions of the CPU budget for processing after classifying features on their feature track length [22]. A different approach is lowering the computational complexity of individual steps involved in the SLAM process [23]. The work in this paper specifically focuses on creating a policy that requires minimum alteration of the underlying framework of the VIO method and proposes an adaptive strategy to alter the parameters according to an online profiling of the resources available.

III. PROPOSED APPROACH

This section details the proposed policy to automatically adapt a set of visual-inertial odometry estimation frameworks such that resilient performance with minimal accuracy degradation is ensured, along with a significant decrease in computational complexity and adaptability to the resources available at any given time during a robot mission. It should be noted that the proposed approach does not exploit factors that are specific to any certain VIO framework, rather a set of ubiquitously applicable methods that could be extended to multiple visual-inertial odometry algorithms are presented.

A. Selected Visual-Inertial Odometry Baselines

The selection of baseline VIO algorithms, VINS-Mono [3], OKVIS [11], and S-MSCKF [5], was made based on two criteria, namely a) their performance on established datasets (e.g., EuRoC [2]), and b) their diversity and utilization of distinct functional principles such that our approach can be applicable to a diverse set of methods. Firstly, while VINS-Mono and OKVIS utilize a non-linear optimisation approach in their back-end with a sliding window in the former and a set of keyframes in the latter, S-MSCKF demonstrates a filter-based approach. Furthermore, while VINS-Mono tracks features detected in images from a monocular camera, OKVIS and S-MSCKF are both stereo based.

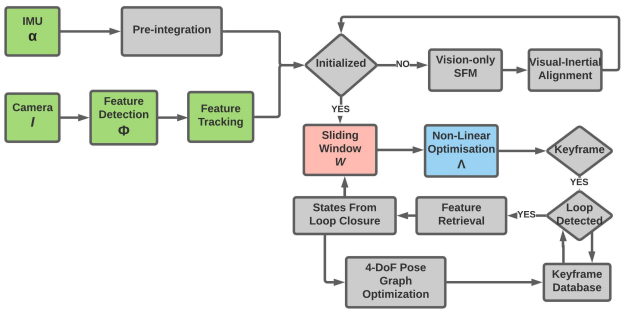

Fig. 1. VINS-Mono block diagram. Modifiable parameters are indicated by the color of the blocks: Green indicates parameter changes during execution, pink indicates parameter changes at compile time, blue indicates setting the parameters prior to execution and grey indicates no changes.

1) VINS-Mono: The "Robust and Versatile Monocular Visual-Inertial State Estimator" (VINS-Mono) [3], is a monocular visual-inertial 6 DoF state estimator based on tightly-coupled sliding window non-linear optimization. Loosely-coupled sensor fusion is performed to initialize the estimator from an unknown initial state. Subsequently, preintegration [24] is performed on IMU measurements before being added to the optimization, and a tightly-coupled formulation for re-localisation is proposed. In the vision processing front-end, for every incoming image frame, robust corner features [25] are detected and tracked using the KLT sparse optical flow algorithm [26]. A minimum separation is enforced between them to ensure uniform feature distribution. The method also features modules for tightly integrated loop closure and 4 DoF pose-graph optimization. Its basic processing steps are illustrated in Figure 1.

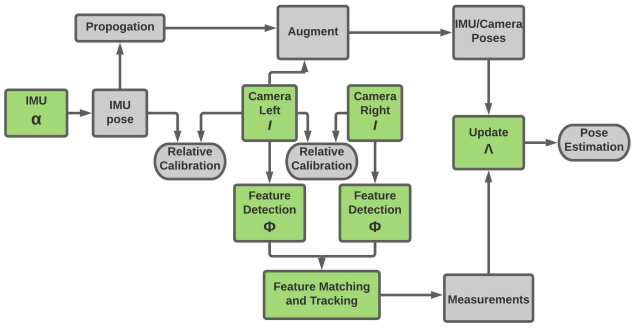

Fig. 2. S-MSCKF block diagram. Modifiable parameters are indicated by the color of the blocks: Green indicates parameter changes during execution while grey indicates no changes.

2) S-MSCKF: The authors of "Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight" [5] contributed S-MSCKF, a stereo, filter-based visual-inertial odometry algorithm that uses the established Multi-State Constraint Kalman Filter (MSCKF) originally proposed in [20]. The original algorithm [20] proposes a measurement model to express the geometric constraints that arise on observing an image feature from multiple camera poses without explicitly adding the features in the state vector. In S-MSCKF, the position of the feature is calculated in the world frame using the least squares method in [20] and the estimated camera

poses. During the filter update step, two camera states are removed and the feature observations obtained are used for the measurement update. The implementation utilizes the FAST [27] feature detector and tracks features temporally using the KLT optical flow algorithm [26] which is also used for stereo feature matching.

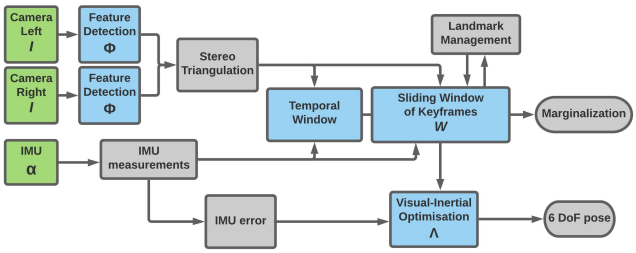

Fig. 3. OKVIS block diagram. Modifiable parameters are indicated by the color of the blocks: Green indicates parameter changes during execution, blue indicates parameter change prior to execution, and grey indicates no changes.

3) OKVIS: The "Open Keyframe-based Visual-Inertial SLAM" (OKVIS) [11] method is based on tightly-coupled fusion of visual information with IMU estimates and uses non-linear optimization over a sliding window of keyframe poses. A probabilistic strategy is used to integrate the IMU error term with the landmark re-projection error resulting in the joint non-linear cost function to be optimized. Partial marginalization of older keyframes in the sliding window is carried-out to maintain a bounded computational complexity. The vision front-end of the algorithm employs a customized Harris corner detector which enforces uniform keypoint distribution and is combined with BRISK descriptor extraction [28]. For initialization, the last pose propagated by IMU measurements is used to obtain a preliminary uncertain estimate of the state.

B. Tracking of Computational Resources

The proposed resource-aware online parameter adaptation of VIO strategies, exploits both prior information for the overall CPU capabilities and online monitoring of CPU core usage during the operation of the robotic system. An initial specification of the CPU that details the number of cores, μ, and the maximum and minimum clock speeds, $\nu_{\max}$ and $\nu_{\min}$, is acquired. While both the core usage and process usage for every process associated with the algorithm are recorded, only the core usage, χ, is used for our parameter adaptation. This information is acquired and updated at a rate of 1Hz using widely available system monitoring tools. When run on multi-core architectures, the core of execution - within the results presented in this work- is fixed to a specified core for ease of resource monitoring.

C. Algorithm Description

Our proposed approach involves altering key parameters of both the vision front-end and the optimization back-end to achieve a reduction in computational resource usage.

1) Front-End Adaptation: The computational reduction in the vision front-end for all methods is achieved by a) choosing to process only a subset, P, of the total incoming frames, I, and b) by changing the maximum number of features, Φ, detected in the frame. The choice of whether frame $I_k \in P$ and thus if is to be processed is determined by an agility metric, α, and the core usage, χ, on which the process is executed. The agility is defined using the average values of linear acceleration, α^a_{avg}, and angular velocity, α^ω_{avg}, over a sliding window of fixed length, l_α. The degree of agility is determined by a comparison with the respective predefined thresholds α^a_T and α^ω_T. We formalize the above in the following equation, where α^ω_n and α^a_n denote the angular velocity and linear acceleration values at time n, while k denotes the time that the latest measurement is received:

$$\begin{aligned} \alpha^\omega_{avg} &= (1/l_\alpha) \sum_{n=k-l_\alpha}^{n=k} \alpha^\omega_n \\ \alpha^a_{avg} &= (1/l_\alpha) \sum_{n=k-l_\alpha}^{n=k} \alpha^a_n \end{aligned} \tag{1}$$

To counter excessive loss of frames in the case of low agility trajectories, a queue over a fixed window of incoming frames is maintained which records whether a frame was processed or dropped and α^ω_T and α^a_T are adjusted based on whether the number of processed frames in the queue, $N(P)$, exceeds or falls below fixed thresholds of minimum, C_0, and maximum, C_1, number of frames to be processed. It is noted that no image frames are dropped during the initialization period.

Finally, the resolution, r, of the images is another factor influencing resource usage and estimation accuracy, and may be altered prior to execution based on the overall computational capabilities at hand.

2) Back-End Adaptation: The computational reduction and adaptability in the back-end is obtained by altering the maximum number of iterations, Λ, required for linearization in case of filter-based methods or by the linear solver for non-linear optimisation based methods. In the case of the latter, the size of the sliding window, W, is also altered.

The alterations made can further be classified into three categories depending on when the changes occur: a) prior to compilation b) at initialization and c) during execution (online) and are highlighted in pink, blue and green respectively in Figures 1-3 for the baseline VIO methods. Certain parameters have a major impact in decreasing resource usage when fixed initially, however have a detrimental impact when they are altered online and lead to instability in the system. In the case of OKVIS, all of these parameters are fixed prior to execution. For S-MSCKF both Φ and Λ are altered online. For VINS-Mono, only Φ is altered online while W and Λ are fixed prior to execution. The aforementioned agility-based selective processing of frames is utilized for all methods and thus the system adapts online which frames to process. On the other hand, the image resolution is fixed prior to execution.

Algorithm 1 outlines the main steps in the proposed approach. Upon initialization, the method provides an initial estimate of all parameters for the VIO method on the basis of the CPU specifications and instantiates a queue, $queue_I$, to record whether the frame I_k is processed or dropped. The choice of altering the maximum number of features detected, Φ, the maximum number of iterations, Λ, and the extent to which they are altered is determined based on a metric $\Delta = \chi - \chi_T$ corresponding to the difference of current core usage and a fixed threshold.

Algorithm 1 Resource-Aware Policy Adaptation

Input: CPU_{spec}, χ_T, $\alpha_T, \kappa_{0,f}, \kappa_{0,p}$

$\Phi, \Lambda, W \leftarrow$ **ComputeInitialParameters**(CPU_{spec})
$\kappa_f \leftarrow 0$, $\kappa_p \leftarrow 0$, $queue_I \leftarrow 0$
for all $I_k \in I$ **do**
  if $(\alpha_k^\omega < \alpha_T^\omega \; and \; \alpha_k^a < \alpha_T^a \; and \; \kappa_f > \kappa_{0,f})$ **then**
    $queue_I.enqueue(0)$, $\kappa_f \leftarrow 0$
    return
  end if
  $\Delta \leftarrow \chi - \chi_T$
  if $(\Delta > \Delta_0 \; and \; \kappa_f > \kappa_{0,f})$ **then**
    $queue_I.enqueue(0)$, $\kappa_f \leftarrow 0$
    return
  else if $(\kappa_p > \kappa_{0,p})$ **then**
    UpdateIterations(Δ, Λ)
    UpdateFeatures(Δ, Φ)
    $\kappa_p \leftarrow 0$
  end if
  $queue_I.enqueue(1)$
  while $(size(queue_I) >= l_I)$ **do**
    $pop \; queue_I$
  end while
  if $(sum(queue_I) > C_0)$ **then**
    Increment α_T^ω, α_T^a
  else if $(sum(queue_I < C_1)$ **then**
    Decrement α_T^ω, α_T^a
  end if
  Increment κ_f, κ_p
end for

Due to different amounts of reduction in the computational resources being achieved by dropping a frame entirely, reducing the number of iterations or reducing the number of features tracked, these adaptations are made on the basis of different thresholds Δ_0, Δ_1 and Δ_2. Additionally, as dropping several frames could lead to an unstable system, it is also governed by the number of frames processed, $N(P)$, over a fixed window. A count of the number of frames since the last frame drop, κ_f, is maintained to ensure that if the current frame is dropped, no frame within $\kappa_{0,f}$ subsequent frames may be dropped. To prevent degeneracy due to rapid changes in Λ or Φ, no parameter is altered till the number of iterations since the last modification, κ_p, exceeds a minimum of $\kappa_{0,p}$ after an alteration is made.

IV. IMPLEMENTATION DETAILS

In this section, we review further details of the particular implementation of our strategy. Tuning certain method-specific parameters apart from those detailed as part of our general approach resulted in better performance however these were not changed while evaluating the algorithm on different sequences of the dataset in the interest of fair evaluation. They were fixed to the values which gave the best performance when tested without the resource-aware algorithm. Loop closure was disabled in every case and all VIO methods were compiled with the highest levels of compiler optimization.

For every VIO algorithm, an initial estimate of the parameters is calculated using the maximum clock speed, ν_{max}, and number of cores in the CPU, μ. The domain for this initialization is divided into three regions, namely R_1 for single core architectures with $\nu_{max} < 1.2$GHz, R_2 for multi-core architectures with $1.2\text{GHz} \leq \nu_{max} \leq 2.1$GHz and R_3 for multi-core architectures with $\nu_{max} > 2.1$GHz. These regions were chosen since the variation of each parameter as a function of the clock speed, ν_{max} and number of cores in the CPU, μ followed a similar trend in the given ranges. Upon experimental evaluation an evident change was noted upon moving between regions.

In VINS-Mono, the size of the sliding window, W, is varied quadratically in R_2, while its value is fixed in R_1 and R_3. The maximum number of iterations, Λ, is varied in steps of two across each region. The number of features, Φ, is varied linearly in all regions.

Since S-MSCKF ensures a uniform distribution of features by ensuring that a certain number of features are detected in every cell of a grid overlayed on the image, the grid dimensions are varied according to the operating region and are fixed prior to execution. The modifications in Φ are linear in R_1 and R_2 and fixed in R_3, while the modifications in Λ are fixed in R_1 and linear in R_2 and R_3. Being a filter-based approach, the size of the sliding window in the backend, W is not applicable.

OKVIS was, by comparison to VINS-Mono and S-MSCKF, computationally expensive and since execution could not be performed on a single fixed core, used two cores instead. Due to OKVIS maintaining a sliding window of IMU-linked temporal frames as well as one of keyframes, two separate window sizes, W_t and W_{kf} were set. It was tested on the x86-based laptop and all parameters were fixed prior to execution with the only online modification being the agility based dropping of frames.

V. EVALUATION STUDIES

The proposed approach is evaluated on the EuRoC [2] dataset in Vicon Room 1 and Vicon Room 2 which includes speeds of upto 0.75m/s along with significant motion blur and differences in illumination. The dataset provides stereo WVGA monochrome images at 20Hz, and temporally synchronized IMU data at 200Hz with ground-truth provided by a VICON motion capture system.

The algorithm is evaluated on the x86 architecture using a dual-core Intel Core i3-4010U CPU with a processor clocked at 1.7 GHz along with a 8GB DDR4 RAM on a laptop, and on the ARM architecture using a quad-core Cortex-A72 (ARM v8) 64-bit SoC at 1.5 GHz processor, with an 8GB LPDDR4 SDRAM on a Raspberry Pi 4B.

The effects of individual parameter variation are calculated by varying a parameter in appropriate step sizes and recording the changes in accuracy and process usage. The best estimate of a parameter is considered to be the value at which the highest accuracy is obtained. Analogously, simultaneous variation is performed to examine evidence of any correlation and combined impact on the accuracy and process usage.

In the optimization back-end of the VIO algorithm, the parameters altered include the number of iterations, Λ, and the window size, W, both of which yield a significant reduction in process usage. A reduction in the former is accompanied with a negligible decline in accuracy while a steep decline is observed on reduction of the latter. In the vision front-end, the number of features, Φ, has the greatest influence on accuracy while the number of frames processed, $N(P)$, has the maximum influence on process usage.

For every trial, the resultant trajectory of each VIO method with and without CRVIOS running alongside was recorded and evaluated using the trajectory evaluation toolbox from [29]. The accuracy is defined by the root mean square error (RMSE) for translation and rotation against ground truth over the entire trajectory. The accuracy and process usage results on the x86-based laptop are shown in Table I and on the ARM-based Raspberry Pi 4B in Table II. The accuracy with the implementation of our policy is comparable to the original algorithm, however a significant reduction in CPU usage is observed. No significant reduction in memory usage was observed.

To evaluate the reaction of the proposed policy to online changes, all cores of the system are artificially stressed for a fixed duration periodically. Indicative plots for the resulting parameter variation in VINS-Mono and S-MSCKF on the ARM-based Raspberry Pi 4B are provided in Figure 4 and Figure 5 respectively, utilizing the EuRoC sequences V1_02 and V2_02. The 3D plots of the corresponding trajectories are provided in Figure 6 and Figure 7 respectively.

Indicative plots comparing OKVIS with and without CRVIOS on EuRoC sequences V1_02 and V2_02 are provided in Figure 8 to illustrate the acceptable reduction in accuracy with the reduction in process usage as can be seen in Table I. A video recording of a selected subset of the results is available at `https://youtu.be/H5gIe418zwc`.

VI. CONCLUSIONS

In this work, we defined a unifying policy that, given different computing systems, provided optimal initial estimates of parameters and varied them according to the online profiling of available computational resources. Our results show that the proposed VIO scheduling functionality achieves almost identical performance to the original methods but with a major reduction in the computational cost. As evaluated

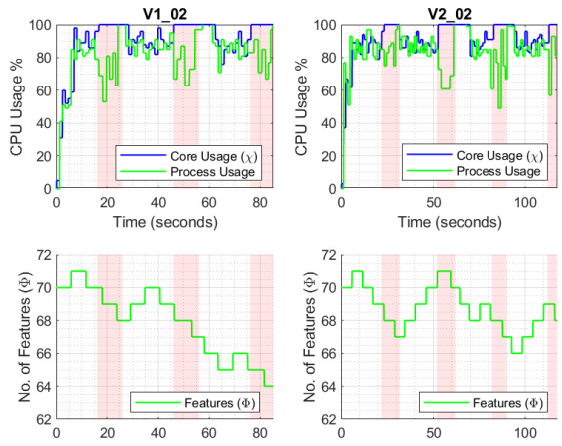

Fig. 4. Parameter adaptation on indicative runs of VINS-Mono on the Raspberry Pi 4B on EuRoC sequences V1_02 (Left) and V2_02 (Right) with periods of artificial stressing shown in red.

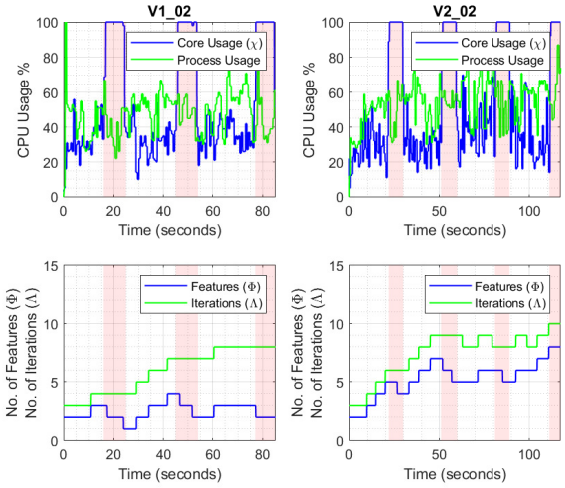

Fig. 5. Parameter adaptation on indicative runs of S-MSCKF on the Raspberry Pi 4B on EuRoC sequences V1_02 (Left) and V2_02 (Right) with periods of artificial stressing shown in red.

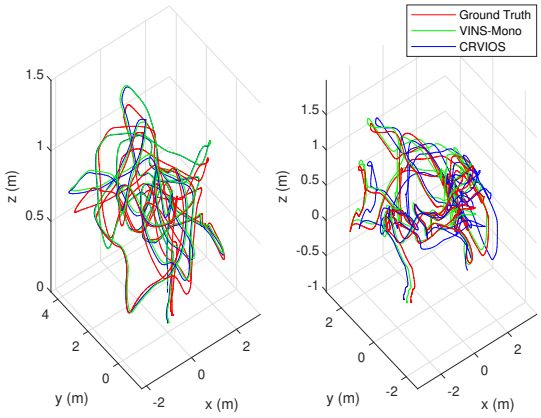

Fig. 6. Indicative plots showing the reduced accuracy on running VINS-Mono with CRVIOS on EuRoC sequence V1_02 (Left) and V2_02 (Right) on the ARM-based Raspberry Pi 4B.

TABLE I

ACCURACY AND PROCESS USAGE CHANGES ON X86 - INTEL CORE I3-4010U @ 1.7GHZ

Dataset	S-MSCKF	CRVIOS		VINS-Mono	CRVIOS		OKVIS	CRVIOS	
	Trans. RMSE (m)		**Change**	**Trans. RMSE (m)**		**Change**	**Trans. RMSE (m)**		**Change**
V1_02	0.179	0.163	0.016	0.156	0.171	-0.015	0.119	0.094	0.024
V1_03	0.176	0.311	-0.135	0.263	0.314	-0.052	0.158	0.180	-0.022
V2_02	0.179	0.243	-0.064	0.160	0.218	-0.058	0.129	0.135	-0.006
V2_03	0.724	0.852	-0.129	0.311	0.334	-0.023	0.184	0.237	-0.053
	Rot. RMSE (deg)		**Change**	**Rot. RMSE (deg)**		**Change**	**Rot. RMSE (deg)**		**Change**
V1_02	1.248	1.615	-0.367	2.203	1.797	0.406	1.196	1.006	0.189
V1_03	2.094	2.377	-0.283	3.042	3.431	-0.389	2.924	3.103	-0.179
V2_02	1.232	2.009	-0.776	2.325	2.557	-0.232	1.021	1.181	-0.160
V2_03	3.441	4.279	-0.838	2.537	2.805	-0.268	1.734	1.514	0.220
	CPU usage %		**Change**	**CPU usage %**		**Change**	**CPU usage %**		**Change**
V1_02	69.7	40.8	28.9	107.6	72.2	35.4	257.0	209.0	48.0
V1_03	64.8	38.6	26.2	96.0	66.9	29.1	247.0	206.0	41.0
V2_02	73.6	42.2	31.4	128.6	77.6	51.0	252.0	223.0	29.0
V2_03	74.5	40.6	33.9	100.9	85.0	15.9	241.0	190.0	51.0

TABLE II

ACCURACY AND PROCESS USAGE CHANGES ON ARM - CORTEX-A72 (ARM V8) @ 1.5GHZ

Dataset	S-MSCKF	CRVIOS	
	Trans. RMSE (m)		**Change**
V1_02	0.162	0.148	0.014
V1_03	0.237	0.223	0.014
V2_02	0.184	0.183	0.001
V2_03	5.274	1.726	3.549
	Rot. RMSE (deg)		**Change**
V1_02	1.464	1.418	0.047
V1_03	2.274	3.155	-0.881
V2_02	1.507	1.472	0.035
V2_03	7.444	3.102	4.342
	CPU usage %		**Change**
V1_02	83.9	63.6	20.3
V1_03	84.2	70.4	13.8
V2_02	97.2	66.9	30.3
V2_03	71.5	57.2	14.3
Dataset	**VINS-Mono**	**CRVIOS**	
	Trans. RMSE (m)		**Change**
V1_02	0.215	0.185	0.030
V1_03	0.257	0.305	-0.047
V2_02	0.163	0.243	-0.080
V2_03	0.311	0.362	-0.051
	Rot. RMSE (deg)		**Change**
V1_02	2.257	2.235	0.022
V1_03	2.983	3.284	-0.301
V2_02	2.615	2.846	-0.231
V2_03	2.537	3.137	-0.600
	CPU usage %		**Change**
V1_02	116.3	100	16.3
V1_03	131.3	90.2	41.1
V2_02	156.6	100.6	56.0
V2_03	156.3	100.6	55.7

on two computationally constrained systems, this improves the ability for their deployment. Although we present the effectiveness of our approach on three state-of-the-art VIO algorithms, it can be easily extended to various VIO systems as it targets common functional blocks in both the vision front-end and optimization back-end. Even though there exists an unavoidable trade-off between the computational cost of a VIO system and its accuracy, our contribution endeavours to ensure that this reduction in performance is minimal.

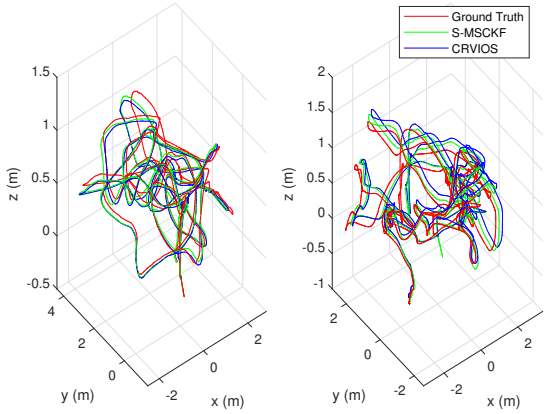

Fig. 7. Indicative plots showing the reduced accuracy on running S-MSCKF with CRVIOS on EuRoC sequence V1_02 (Left) and V2_02 (Right) on the ARM-based Raspberry Pi 4B.

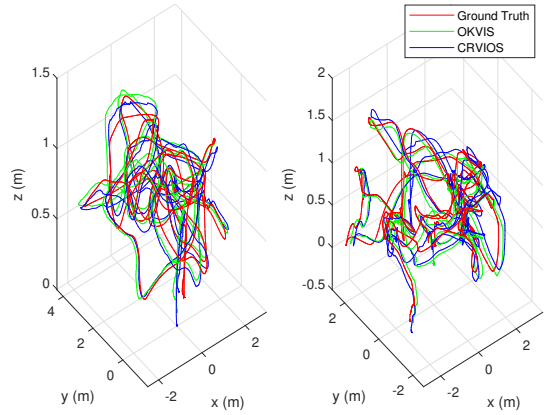

Fig. 8. Indicative plots showing the reduced accuracy on running OKVIS with CRVIOS on EuRoC sequence V1_02 (Left) and V2_02 (Right) on the x86-based laptop. The corresponding reduction in process usage can be seen in Table I.

REFERENCES

[1] M. Achtelik, M. Achtelik, Y. Brunet, M. Chli, S. Chatzichristofis, J. Decotignie, K. Doth, F. Fraundorfer, L. Kneip, D. Gurdan, L. Heng, E. Kosmatopoulos, L. Doitsidis, G. H. Lee, S. Lynen, A. Martinelli, L. Meier, M. Pollefeys, D. Piguet, A. Renzaglia, D. Scaramuzza, R. Siegwart, J. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss, "Sfly: Swarm of micro flying robots," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 2649–2650.

[2] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract

[3] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. PP, 08 2017.

[4] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4666–4672.

[5] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, 2018.

[6] S. Khattak, C. Papachristos, and K. Alexis, "Keyframe-based direct thermal–inertial odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2019.

[7] S. Khattak, F. Mascarich, T. Dang, C. Papachristos, and K. Alexis, "Robust thermal-inertial localization for aerial robots: A case for direct methods," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2019, pp. 1061–1068.

[8] S. Zhao, P. Wang, H. Zhang, Z. Fang, and S. Scherer, "Tp-tio: A robust thermal-inertial odometry with deep thermalpoint," *arXiv preprint arXiv:2012.03455*, 2020.

[9] C. Campos, R. Elvira, J. J. Gomez, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," *arXiv preprint arXiv:2007.11898*, 2020.

[10] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[11] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015. [Online]. Available: https://doi.org/10.1177/0278364914554813

[12] Z. Zhang, A. Suleiman, L. Carlone, V. Sze, and S. Karaman, "Visual-inertial odometry on chip: An algorithm-and-hardware co-design approach," in *Robotics: Science and Systems*, 2017.

[13] A. J. Davison and D. W. Murray, "Simultaneous localization and map-building using active vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 865–880, 2002.

[14] H. Strasdat, C. Stachniss, and W. Burgard, "Which landmark is useful? learning selection policies for navigation in unknown environments," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 1410–1415.

[15] G. Huang, M. Kaess, and J. J. Leonard, "Consistent sparsification for graph optimization," in *2013 European Conference on Mobile Robots*, 2013, pp. 150–157.

[16] L. Carlone, Z. Kira, C. Beall, V. Indelman, and F. Dellaert, "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4290–4297.

[17] A. Fontán, J. Civera, and R. Triebel, "Information-driven direct rgb-d odometry," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4928–4936.

[18] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234, 2007.

[19] K. Konolige and M. Agrawal, "Frameslam: From bundle adjustment to real-time visual mapping," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1066–1077, 2008.

[20] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3565–3572.

[21] M. Li and A. I. Mourikis, "Vision-aided inertial navigation for resource-constrained systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1057–1063.

[22] D. G. Kottas, R. DuToit, A. Ahmed, C. X. Guo, G. Georgiou, R. Li, and S. Roumeliotis, "A resource-aware vision-aided inertial navigation system for wearable and portable computers," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[23] L. M. Paz, J. D. Tardos, and J. Neira, "Divide and conquer: Ekf slam in o(n)," *Trans. Rob.*, vol. 24, no. 5, p. 1107–1120, Oct. 2008. [Online]. Available: https://doi.org/10.1109/TRO.2008.2004639

[24] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[25] Jianbo Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[26] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, p. 674–679.

[27] M. Trajkovic and M. Hedley, "Fast corner detection," *Image and Vision Computing*, vol. 16, pp. 75–87, 02 1998.

[28] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *2011 International Conference on Computer Vision*, 2011, pp. 2548–2555.

[29] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.