OASIS: Optimized Adaptive System for Intelligent SLAM

ALLES REBEL, SDSU/UCI Computational Science Research Center, USA
NIKIL DUTT, Department of Computer Science, University of California Irvine, USA
BRYAN DONYANAVARD, Department of Computer Science, San Diego State University, USA

Visual Simultaneous Localization and Mapping (VSLAM) is essential for mobile autonomous systems operating in complex dynamic environments. VSLAM algorithms are computationally intensive and must execute in real-time on resource-constrained embedded devices. Variations in environmental complexity can lead to longer frame processing times, causing dropped frames, lost localization information, and degraded accuracy. To address these challenges, we introduce OASIS, a novel adaptive approximation method that dynamically reduces input frame areas based on realtime visual importance. Unlike traditional optimizations that require adjusting internal SLAM parameters, OASIS selectively minimizes computation by adaptively filtering less critical image regions, significantly reducing computational load. Evaluations on the EuRoC MAV dataset demonstrate that our approach balances accuracy and system predictability, achieving up to a 71.8% reduction in worst-case pose estimation errors. OASIS offers a significant advancement in reliable, predictable, and energy-efficient SLAM tailored for mobile autonomous robotic applications.

CCS Concepts: • **Applied computing**; • **Computer systems organization → Embedded software**; *Robotic autonomy*; *Robotic components*;

Additional Key Words and Phrases: Visual SLAM, Adaptive Masking, Realtime Processing, Embedded Systems, Pose Estimation, Computational Efficiency, Autonomous Navigation, ORBSLAM3

ACM Reference Format:
Alles Rebel, Nikil Dutt, and Bryan Donyanavard. 2025. OASIS: Optimized Adaptive System for Intelligent SLAM. *ACM Trans. Embedd. Comput. Syst.* 1, 1, Article 1 (January 2025), 22 pages. https://doi.org/10.1145/3761808

1 Introduction

Autonomous cyber-physical systems (A-CPS) execute complex software pipelines on resource-constrained devices, for applications that interact with the physical world without human intervention. A-CPS must collect and process sensory information in order to understand and navigate the environment. Visual-inertial simultaneous localization and mapping (SLAM) is an essential task for A-CPS. State-of-the-art systems such as ORBSLAM3 [4] have demonstrated that robust, realtime localization and mapping are achievable without external infrastructure. This capability is particularly vital for small-scale embedded platforms, where weight, power, and computational resources are at a premium, making expensive sensors like LiDAR impractical.

Modern research has focuses on adapting SLAM algorithms to operate under the constraints of micro-scale platforms. For instance, micro aerial vehicles (MAVs) have successfully executed visual-inertial SLAM onboard by leveraging lightweight cameras and optimized algorithms designed for low-power processors [7, 14]. Fig 1 illustrates how integrating

Authors' Contact Information: Alles Rebel, arebel4097@sdsu.edu, SDSU/UCI Computational Science Research Center, San Diego, California, USA; Nikil Dutt, Department of Computer Science, University of California Irvine, Irvine, California, USA, dutt@uci.edu; Bryan Donyanavard, Department of Computer Science, San Diego State University, San Diego, California, USA, bdonyanavard@sdsu.edu.

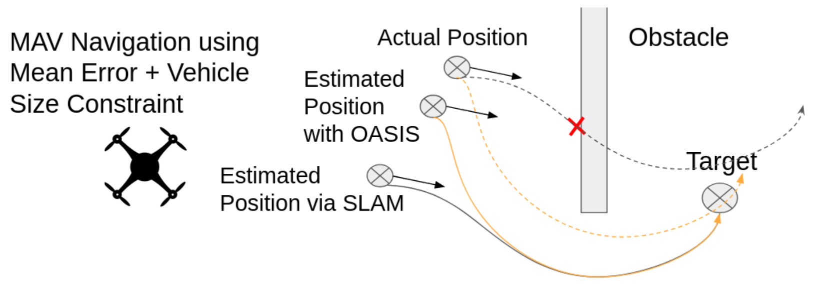

MAV Navigation using Mean Error + Vehicle Size Constraint
Actual Position
Obstacle
Estimated Position with OASIS
Estimated Position via SLAM
Target

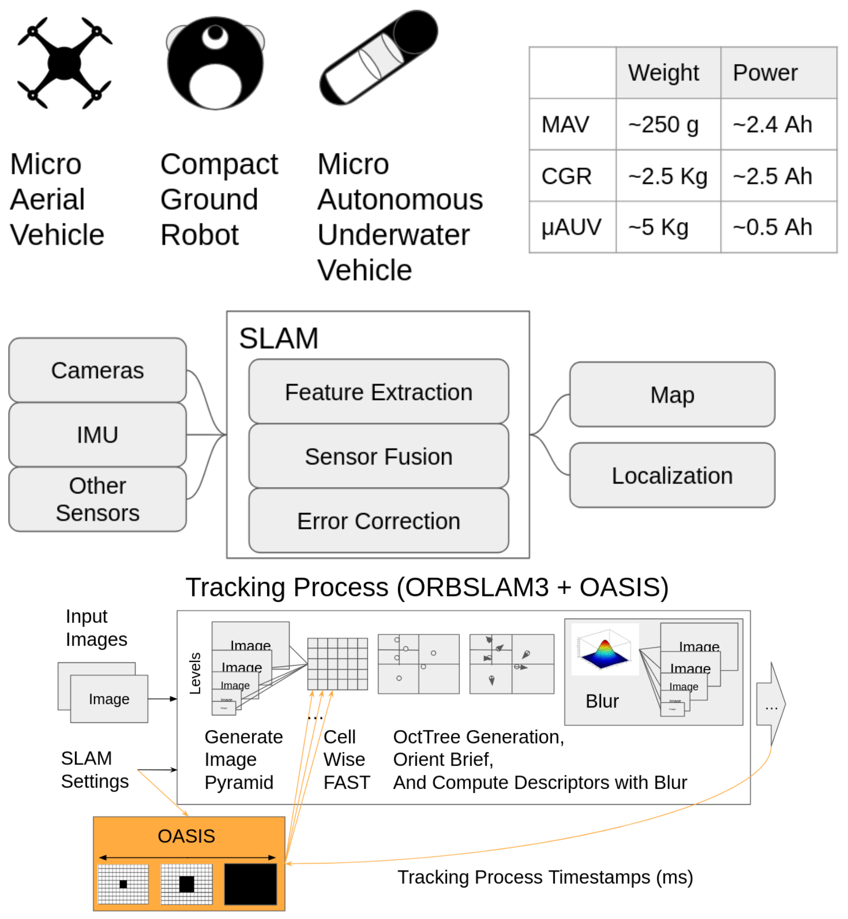
Fig. 1. Illustration of a worst-case scenario in autonomous trajectory planning. The figure compares two route plans: one based on standard SLAM, and the other using SLAM enhanced with OASIS. Dark arrows show velocity vectors, while curved lines represent the planned trajectories. The orange route, from the OASIS-enhanced SLAM, shows a 70% improvement in worst-case estimation accuracy. When both routes are applied to the true position (shown as dotted lines), the standard SLAM route indicates a potential collision with nearby obstacles.

visual and inertial sensor data in SLAM not only enhances route planning accuracy, but also mitigates collision risks in a MAV scenario. These approaches exploit commodity sensor technology to achieve realtime performance while keeping costs and payload weights minimal, which is essential for autonomous missions in inaccessible or hazardous environments [4, 18]. Similarly, compact ground robots have been equipped with embedded visual SLAM systems that utilize algorithmic simplifications to perform robust localization on limited computational hardware [9]. In the underwater domain, research on micro autonomous underwater vehicles (μAUVs) has shown that alternative sensing strategies (e.g., acoustic or electromagnetic localization) can be integrated with visual and inertial data to enable SLAM in challenging, low-visibility conditions [5]. These works underscore a common theme: by balancing sensor choice and algorithmic efficiency, it is possible to overcome the inherent limitations of small, low-power platforms (Fig 2). Despite these advances, the implementation of real-time SLAM on embedded platforms continues to present formidable challenges. Key issues include limited computational capacity, energy budget, system predictability and robustness. In this work, we define robustness as the system's capability to consistently maintain a stable mean Absolute Trajectory Error (ATE) despite variability and uncertainty in environmental conditions, computational loads, or sensor inputs. A robust system maintains a consistently low mean ATE, indicating stable performance across typical operating conditions. Further, for this work we define predictability explicitly as the system's ability to minimize the maximum observed ATE (Max ATE), ensuring that worst-case errors remain bounded and manageable. Predictability, measured via Max ATE or worst case ATE, directly supports reliable route planning and safe autonomous operation, as systems can confidently account for and mitigate potential worst-case localization errors.

1.1 Motivation and Problem Statement

A-CPS provide clear advantages: they can operate in environments too hazardous or remote for direct human presence, and they can do so persistently without fatigue. A-CPS applications rely on accurate realtime SLAM to provide environmental understanding. ORBSLAM3 is a state-of-the-art visual-inertial SLAM application that offers the accuracy and robustness needed for such missions [4]. However, deploying ORBSLAM3 on autonomous devices in practice faces serious challenges, primarily stemming from real time and embedded constraints [20].

In this work, we address performance degradation experienced by realtime SLAM executing on resource-constrained autonomous devices. Unlike standard desktop or lab settings, an embedded autonomous device must perform localization



	Weight	Power
MAV	~250 g	~2.4 Ah
CGR	~2.5 Kg	~2.5 Ah
µAUV	~5 Kg	~0.5 Ah

Fig. 2. Top: Example of a Micro Aerial Vehicle (MAV), a Compact Ground Robot (CGR), and Micro Autonomous Underwater Vehicle (μAUV). Each of these robots integrates core sensor suites—cameras, IMUs, and other sensors for very different mission environments. The table highlights key differences in weight and power usage, with examples based on real-world systems: DJI Mavic Mini (MAV), iRobot's J7 Roomba (CGR), and General Dynamics Bluefin SandShark (μAUV).
Bottom: A detailed block diagram of a ORBSLAM3 based perception pipeline with OASIS augmenting the processing of cells in the ORB Featurizer of the Tracking Process. OASIS provides the signal to process a specific cell of the image pyramid through maskin. OASIS recieves timestamp information from the Tracking Map update phase and from the SLAM setting of the entire pipeline.

and mapping within strict timing and hardware limitations. In particular, we identify two compounding issues that can drastically affect SLAM performance on an autonomous device, in addition to timing and hardware limitations:

- **Realtime Responsiveness:** SLAM pipelines are soft realtime systems – they do not hard-fail if a deadline is missed, but the consequences of lag are severe. Studies have shown that if frames are not processed as fast as they arrive, the system exhibits significant localization drift [20]. Another issue lag may cause is tracking loss, which triggers a costly recovery procedure (relocalization) that pauses map expansion [20]. Both can lead to a collision due to planning or other processes relying on inaccurate or inconsistent localization.

- **Worst-Case Performance:** Deadline misses have an outsized affect on worst-case performance. Any information extracted from these dropped or skipped frames yields a dramatic improvement in worst-case performance. Having a localization and map that is consistent is critical for route or mission planning processes in the perception pipeline [16].

ORBSLAM3 deployed on an embedded system must cope with tradeoffs: the need for high-performance vision processing, and the reality of limited onboard computing capability. This paper tackles this tradeoff by systematically analyzing ORBSLAM3's behavior under realtime embedded conditions and quantifying the tradeoffs between frame rate, computation, and accuracy. We specifically investigate how frame deadline misses and dropped frames affect localization and mapping error in worst-case scenarios, and we highlight techniques to improve the realtime robustness and predictability of SLAM on devices with limited computational capabilities.

By explicitly understanding and addressing the challenges of realtime responsiveness, computational constraints, robustness, and predictability, our objective is to facilitate fully autonomous systems that reliably execute missions in demanding operational scenarios. In particular, we aim to mitigate the adverse effects of localization drift, pose estimation errors, and unpredictable worst-case performance—issues that currently hinder the deployment of state-of-the-art Visual-Inertial SLAM on embedded autonomous platforms. Overcoming these barriers is crucial for enabling robust autonomous capabilities, such as precision infrastructure inspection, accurate environmental monitoring, and effective search-and-rescue operations, with Visual-Inertial SLAM serving as the reliable foundation of the perception pipeline.

1.2 Contributions of This Work

This paper introduces OASIS, a novel optimized adaptive approximation method specifically tailored for ORBSLAM3 in embedded realtime environments. Our primary contributions are:

- Addressing the constraints of resource-limited embedded processors, we propose Region-Specific Approximation via masking. This strategy prioritizes the processing of stable, centrally located features critical for robust keyframe selection and map consistency. Our approach improves the efficiency of the system while maintaining base SLAM accuracy (mean ATE).
- We propose a fully online, runtime adaptive masking strategy that selectively processes only the most informative regions of each input frame based on the current computational budget and timing constraints. As a result, the mechanism prevents frame dropouts and maintains consistent tracking performance (low max ATE).
- We implement and instrument experiments using ORBSLAM3, the EuRoC MAV dataset, and a representative embedded platform that show our method method reduces worst-case Absolute Trajectory Error (ATE) by **71.8%** and, in several scenarios, also improves the mean ATE.

2 Background and Related Works

Simultaneous Localization and Mapping (SLAM) is the process of constructing a map of an unknown environment while concurrently estimating an agent's position within that environment. SLAM systems typically integrate data from sensors such as LiDAR, cameras, and Inertial Measurement Units (IMUs) to collect environmental information. A crucial step in this process is featurization: the extraction of distinctive features from raw sensor data. Featurization helps identify and match landmarks in different frames. Popular featurization techniques include Scale-Invariant Feature Transform (SIFT) and Oriented FAST and Rotated BRIEF (ORB), both of which are widely used due to their robustness and efficiency [11, 19]. To manage uncertainties and correctly associate sensor measurements with these extracted features, sensor fusion methods like the Extended Kalman Filter (EKF) are often employed to remove error, resulting in visual inertial SLAM. Loop closure detection, which involves recognizing previously visited locations, combined with graph-based optimization techniques, helps to minimize cumulative errors and refine the final map. Bag of Words approaches, inspired by natural language processing, further assist in determining if a particular set of feature descriptors corresponds to the same physical location [6].

2.1 Feature Extraction in SLAM

Feature extraction identifies distinctive landmarks (features) within sensor data, typically images, to facilitate consistent matching across frames. These extracted features form the basis of pose estimation and mapping in visual SLAM. Popular visual feature extraction methods include Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), and Oriented FAST and Rotated BRIEF (ORB) [12, 19]. A robust feature extractor consistently identifies features despite changes in viewpoint, illumination, and image scale, enabling accurate localization and motion estimation across frames.

2.2 Keyframe Graph-Based SLAM

Graph-based SLAM represents a significant advance over earlier filter-based methods by selectively processing a subset of frames known as keyframes. Keyframes are frames that capture significant changes or novel information about the environment. Instead of incorporating every frame, the SLAM system uses keyframes as nodes within a pose graph. Edges in this graph represent spatial-temporal constraints (relative pose measurements) between keyframes. Graph-based optimization algorithms, such as pose graph optimization, incrementally refine the poses of keyframes, maintaining global consistency in the presence of loop closures and sensor noise. Robust loop closure detection is especially critical in keyframe-based SLAM to mitigate drift and maintain an accurate and consistent global map over extended periods of operation.

2.3 ORBSLAM3

ORBSLAM3 is a state-of-the-art implementation of a visual inertial SLAM that integrates ORB feature extraction with keyframe graph-based optimization. With consistent ORB features, the system efficiently estimates motion and constructs a graph of keyframes as nodes and their common features as edges [4]. ORBSLAM3's powerful loop closure detection based on Bag of Words place recognition mitigates drift accumulation on detection. This makes ORBSLAM3 an optimal choice for applications where precise, realtime mapping and localization are critical.

When applying ORBSLAM3 to EuROC dataset on a desktop class processor, the entire perception pipeline has no issue completing all tasks prior to new data being fed into the pipeline. In other words, the desktop class processor performs in a realtime manner by default due to the compute capacity. The same experiment on an embedded processor results in frame times that quickly exceed the timestamp of the next frame being fed into the pipeline (see Table 2). If frame processing times were on average the same as the incoming framerate, a queue to control the flow rate would serve as a potential solution with delay being added into the system. However, in the context of an embedded processor, this would result in frames being dropped in favor of fresher data. For this work, realtime will be defined as processing all incoming data prior to the next expected delivery of data. We will further focus on processing of video data, due to the computational complexity.

2.4 Related Works

Methodologies to enable embedded devices to utilize State-of-the-art SLAMs typically address computational deadlines through full-frame adjustments. For example, systems may alter resolution, frame rate [2], limit features per cell [15], or adjust configurations at runtime (e.g., SlimSLAM [1], DOG-SLAM [10]). These methods perform an offline design space search, utilizing metrics that extend beyond those available during operational trials (e.g., using signals such as absolute error to determine which configuration to adjust for optimal performance). They correlate these findings from offline trials to metrics available online during operation. Although effective, these methods still apply uniform processing across the entire frame or pipeline, overlooking opportunities for region-specific optimizations such as adaptive frame processing. This highlights the potential for localized processing strategies better suited to resource-constrained platforms.

Some approaches leverage expensive hardware accelerators, such as GPUs or dedicated neural processing units, to achieve realtime performance on embedded systems [13]. However, these accelerators are often shared among multiple processes such as object detection, labeling, and other tasks within the perception pipeline [17], which can lead to contention in memory bandwidth or computational resources. Such resource sharing may result in system failures if not properly managed [8].

Our work addresses these limitations by integrating targeted, region-specific approximation techniques within a completely online framework, using only resources allocated for mapping and localization in a standard perception pipeline (CPU only). We only utilize online-only metrics to apply our approximation techniques. Our approach ensures resource allocation with adherence to strict computational deadlines in deployment of state-of-the-art SLAM on embedded mobile devices.

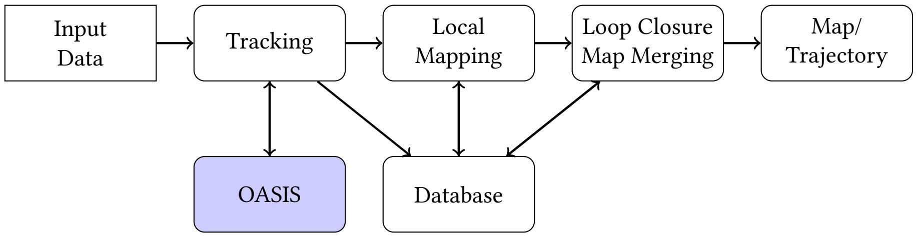


Fig. 3. ORBSLAM3's overall structure: an Input Data block (Images, IMU) connects to a Tracking block; Tracking feeds into Local Mapping and OASIS blocks; Local Mapping connects to Loop Closure and Database blocks; Loop Closure links to both the Database and Map/Trajectory blocks; bidirectional arrows indicate feedback loops. Rounded blocks indicate SLAM processes. Sharp blocks indicate information external to SLAM operation. Blue block indicates our method.

3 Optimized Adaptive System for Intelligent SLAM (OASIS)

To facilitate computationally bound pose estimation tasks, we propose an Optimized Adaptive System for Intelligent SLAM (OASIS). OASIS is a dynamic masking strategy, where the computation is bound by real time constraints and the resulting accuracy based on a processing budget. Our method predicts the available processing time for the next frame using a moving average of local mapping times and leverages the timing and SLAM settings information to decide which cells to enable or skip during the feature extraction stage of the perception pipeline. Fig 3 illustrates how the proposed OASIS module (highlighted in blue) is integrated within the ORB-SLAM3 pipeline.

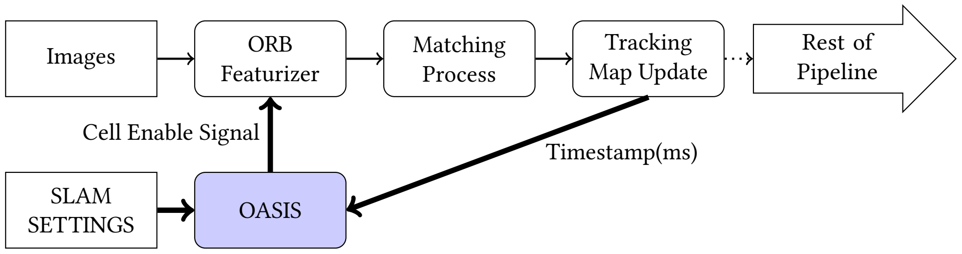


Fig. 4. OASIS integration into the ORBSLAM3 pipeline. Timing from tracking map update and SLAM settings is used to generate a binary signal that enables specific cells for processing in the ORB Featurizer, optimizing feature extraction under computational constraints. Rounded blocks indicate SLAM processes. Sharp blocks indicate information external to SLAM operation. Blue block indicates our method.

3.1 Dynamic Masking Budget Estimation

To balance efficiency with pose estimation accuracy, our system dynamically adjusts the mask size by following a multi-stage process. Both the data produced by the ORB-SLAM3 pipeline and consumed by our budget predictor, as well as the corresponding control signals transmitted to the ORB-SLAM3 featurizer, are shown in Fig 4. We estimate the computational budget for the next frame by computing a moving average of number of cells processed per frame:

$$\bar{C}(n) = \frac{1}{n} \sum_{i=1}^{n} C(i), \tag{1}$$

where $C(i)$ denotes the number of processed cells for frame i, and n is the current frame index. Using the actual processing time $T_{actual}(n)$ of the most recent frame, we estimate the average time required per cell as:

$$t_{cell}(n) = \frac{T_{actual}(n)}{\bar{C}(n)}. \tag{2}$$

Given a total allowed processing time per frame T_{frame}, the available computational budget in terms of cell count for the next frame is calculated as:

$$B_{frame}(n+1) = \frac{T_{frame}}{t_{cell}(n)}. \tag{3}$$

If operating in stereo mode (two images per frame), this budget is halved to account for increased computational load.

3.1.1 Dynamic Mask Size Determination. The optimal mask size for the next frame is determined by iteratively evaluating candidate mask sizes. For a given candidate mask size m, the number of cells covered across all pyramid levels is computed as:

$$C(m) = \sum_{l=0}^{L-1} \min\left(N_l,\, m^2\right), \tag{4}$$

where N_l is the total number of cells at pyramid level l and L is the number of pyramid levels. Starting from a minimum mask size (e.g., $m = 2$), the algorithm increases m until $C(m)$ meets or exceeds the frame budget B_{frame}. The selected mask size is then given by:

$$M(n+1) = m^*, \tag{5}$$

where m^* is the largest candidate for which $C(m^*) < B_{frame}$ (with the next size, $C(m^* + 1)$, exceeding the budget). Practically, masking algorithm comprises a 13% to 100% approximation of the input original frame depending on the budget.

3.2 Masking Candidates

In our system, the dynamic masking strategy not only adapts to the available processing budget but also takes into account conditions that have been observed to contribute significantly to SLAM error reduction—primarily those captured in keyframes. Keyframe selection is based on several criteria, and these criteria naturally suggest a masking strategy that grows from the middle outwards.

Points in the center of an image tend to move less between frames compared to points near the edges. This lower relative motion leads to more consistent tracking over time. Even if these central points are not ideal in every respect (for example, they may sometimes correspond to farther objects), their reliability means that processing them still yields valuable information for SLAM. This concept is closely linked to our keyframe selection and removal criteria, as described below.

3.2.1 Keyframe Selection. Keyframe insertion is governed by several criteria, including temporal gaps, local mapping status, and the quality of tracked map points. The stability of central points contributes to these criteria in the following ways:

- **Temporal Stability:** Because central points exhibit slower motion, they remain consistently tracked over multiple frames. This stability helps ensure that the system can accurately compare frames over time, making it easier to detect when enough new information has been acquired to warrant a new keyframe.
- **Reliable Map Point Observations:** The persistence of central features—even if not optimal for depth estimation—provides a reliable baseline of map points. This is crucial when evaluating conditions such as the number of tracked map points in the reference keyframe versus the current frame, a key factor in the decision process for keyframe creation.
- **Robustness in Challenging Conditions:** In scenarios where peripheral features may move rapidly or become occluded, the central features maintain consistency. This consistency allows the system to confidently trigger keyframe insertion when the overall quality of the tracked features meets the established criteria.

The observation that points in the center of an image tend to exhibit slower relative motion between frames directly motivates our use of a centralized mask. By concentrating processing on the central region, our system leverages the reliably tracked features - even when the full frame cannot be processed due to time constraints imposed by complex scenes. Although this approach may sacrifice peripheral scene perception, it ensures robust SLAM performance by focusing on the most stable spatial information.

This principle is closely linked to the ORBSLAM3 keyframe selection criteria. The central features, which are consistently observed, provide a dependable basis for new keyframe insertion. Specifically, the temporal thresholds and mapping conditions used for keyframe creation are designed to exploit the high reliability of central points, ensuring robust and continuous map point observations. Although central points are not optimal from a depth perspective, their consistent tracking facilitates accurate pose estimation.

The redundancy of these stable central features is leveraged during keyframe removal. A keyframe is considered redundant when a large proportion (e.g., 90%) of its map points are observed in at least three other keyframes. Since central points are persistently tracked, they are more likely to appear in multiple keyframes, and this redundancy is exploited to cull non-essential keyframes. The additional choices allow for an better map representation than the realtime scenario.

3.2.2 Keyframe Removal. The keyframe culling strategy aims to remove redundant keyframes, those where a high percentage of the map points (often from the central, stable region) are observed in other keyframes. The reliability of central points factors into keyframe removal as follows:

- **High Redundancy of Central Points:** Since central features are tracked reliably across multiple frames, they tend to be present in several keyframes. If 90% of the map points in a keyframe (primarily central points) are also observed in at least three other keyframes, that keyframe is deemed redundant.
- **Limited Novelty:** Keyframes that mostly contain these well-tracked central points might not offer additional useful information compared to neighboring keyframes. This overlap supports the culling process by identifying keyframes that can be safely removed without compromising the robustness of the SLAM map.

3.3 Cell Manager Implementation

The cell manager module, implemented in C++ and integrated with ORBSLAM3, leverages the above budget estimation to control cell processing. Its main functions include:

- **Cell Counting and Skipping:** As each cell is processed, it is counted. Cells are selectively skipped if they fall outside the current Field-of-View (FOV) mask or if frame skipping is required.
- **Frame Budget Adjustment:** The module calculates the time per cell using the actual frame time and the average cells per frame (see Eq. 2). It then computes the frame budget in terms of cell count (Eq. 3). If the actual frame time exceeds T_{frame}, the system compensates by skipping frames and recalculating the budget.
- **Iterative Mask Selection:** Starting with a 2×2 mask, the cell manager iteratively increases the mask size. For each candidate mask size, it computes the number of cells covered across pyramid levels (Eq. 4). When the cumulative count exceeds the budget, the previous mask size is selected as the optimal FOV mask (Eq. 5).

Pseudo code for the above procedures is provided as Algorithm 1.

4 Experimental Setup

In this work, we focus on evaluating two critical performance metrics for visual-inertial SLAM systems: the reduction of worst-case pose estimation errors (maximum error) and the preservation of system responsiveness by consistently meeting realtime deadlines.

Our evaluation is carried out on the widely recognized EuRoC micro aerial vehicle datasets [3]. In particular, we use eleven datasets that capture a diverse range of environmental conditions and motion dynamics typical of high-speed pose estimation tasks. The EuRoC datasets provide synchronized stereo images, inertial measurements, and high-precision ground truth, making them ideal for testing SLAM algorithms. These datasets are divided into two batches: one recorded in an industrial machine hall with millimeter accurate position ground truth from a laser tracker; and another captured in a Vicon equipped room, offering 6D pose ground truth and detailed 3D reconstructions from a laser scanner.

For each dataset, ten independent trials are executed using the default ORBSLAM3 configuration (Table 6) for the EuRoC datasets. The scenarios are designed to facilitate a comparative analysis between our proposed OASIS method and a realtime baseline, with a particular emphasis on pose estimation accuracy and system responsiveness. The realtime baseline is defined as the standard ORBSLAM3 configuration running without any adaptive computational adjustments, operating strictly under the constraint that frames must be processed within their inter-arrival times. Frames exceeding this constraint are dropped without processing, ensuring strict adherence to realtime constraints. Furthermore, we compare our method to state-of-the-art adaptation methodologies.

We quantify improvements in pose estimation accuracy by measuring both the mean and maximum absolute errors across all trials. Specifically, the experiments are designed to capture the reduction of worst-case errors by comparing error metrics between the baseline and OASIS implementations. Additionally, we track the system's responsiveness by monitoring adaptive adjustments in mask sizes, which dynamically balance computational efficiency with tracking accuracy, and tracking processing time of the perception pipeline processes.

4.0.1 Baseline Performance under Unconstrained Computation. To establish an understanding of impact in the absence of computational constraints, all experiments are repeated on an Intel i7-6950X host system to replicate the computing environment originally employed by the authors of ORB-SLAM3 [4]. Detailed specifications for both systems used for evaluation are provided in Table 1.

4.0.2 Impact Analysis of Varying Mask Sizes. In addition, we conduct systematic experiments to characterize how SLAM performance is influenced by variations in mask size. These evaluations involve running the SLAM pipeline across the entire dataset under a range of fixed mask sizes. This investigation supports analysis of the accuracy and computational efficiency trade offs inherent in dynamic mask sizing decisions.

4.0.3 Comparative Evaluation against Existing Adaptive Methods. Beyond assessing performance relative to a standard realtime baseline (i.e., ORB-SLAM3 with default parameters while considering deadlines), our method is benchmarked against established adaptive SLAM techniques and results from the non-adapative fixed mask trials:

- **PID SLAM [15]:** An adaptive method employing a modified Proportional-Integral-Derivative (PID) controller, dynamically adjusting the number of extracted features based on observed pose variations.
- ω **SLAM [2]:** A method adaptively performing frame skipping, driven by pose, both rotation and translation changes.
- **Fixed Mask Size (Non-adaptive Baseline):** A control approach maintaining a constant mask size throughout all experimental trials.

4.0.4 Isolated Evaluation of Dynamic Budget Estimation under Computational Load. To evaluate the efficacy of our dynamic budget estimation strategy, we perform an isolation experiment with controlled computational stress. Periodic load was artificially imposed on each processor core using the *stress-ng* utility, configured to induce a computational load at a consistent 10% duty cycle for a period of 10 seconds. Three distinct configurations are compared under these conditions:

(1) **ORB-SLAM3 Default Configuration** Operates without explicit consideration of computational constraints or deadlines.
(2) **ORB-SLAM3 with Realtime Constraints** This is the realtime baseline discussed prior.
(3) **OASIS (Dynamic Masking with Budget Estimation)** Our dynamic budget estimation to adapt mask sizes in realtime, explicitly managing computational resources in the presence of periodic induced compute contention.

All experiments are conducted on NVIDIA Jetson Orin NX development kit and an Intel i7-6950X desktop computer. The Nvidia Jetson is representative of hardware that could be used in mobile and autonomous systems. The desktop computer is representative of a host system the authors of ORBSLAM3 used to evaluate their SLAM implementation [4].

The software environment is built upon ORBSLAM3 [4], augmented with an adaptive mask sizing mechanism (OASIS). ORBSLAM3 only takes advantage of vectorized commands offered by the CPU, through OpenCV. In other words, ORBSLAM3 is run in its default CPU-only configuration, no enhancements are done to offload to GPU. The open

Specification	**Details**
Jetson Orin NX 16GB Developer Kit	
CPU	8-core Arm Cortex-A78AE
Memory	16 GB 128-bit LPDDR5
Power Limit	25W
Host Machine	
CPU	10-core Intel Core i7-6950X
Memory	64 GB 256-bit DDR4
Power Limit	140W

Table 1. Hardware Specifications for Jetson and Host Machine Used in Experiments

source implementation C/C++ leverages multi-threaded processing to concurrently handle feature extraction and pose estimation, eigen for matrix operations, and g2o for graph optimization.

5 Results & Discussion

Table 2 provides an aggregated comparison of error metrics between the realtime baseline and the OASIS method, averaged over all datasets. Our method demonstrates a significant reduction in both average maximum and average mean absolute errors, with the maximum error reduced by 62.1% and the mean error by 30.8%. These improvements validate our approach's ability to drastically mitigate worst-case errors while maintaining stable overall performance, crucial for realtime embedded applications. Table 2 details the performance on individual datasets. Notably, while the mean absolute error remains largely consistent, dramatic improvements in maximum error (e.g., MH03 exhibits a drop from 1.50386 m to 0.12712 m) confirm that our approach primarily mitigates outlier error cases. This effect is essential in applications where sporadic large errors can have severe consequences.

Jetson Orin NX 16GB Developer Kit											
Dataset	**Realtime Baseline**			**OASIS**		**Mean ATE (m)**		**Max ATE (m)**		**Improvement (%)**	
	Frames	**Dropped (%)**	**FPS**	**Mask (Mean ± Std)**	**Dropped (%)**	**Realtime**	**OASIS**	**Realtime**	**OASIS**	**Mean ATE**	**Max ATE**
MH01	3682	943.1 (25.61%)	15.83	12.23 ± 4.51	**0.0 (0.00%)**	**0.06805**	0.07001	0.27469	**0.18119**	**-2.9%**	**34.0%**
MH02	3040	848.9 (27.92%)	15.36	14.37 ± 5.38	**0.0 (0.00%)**	0.07195	**0.04817**	0.33743	**0.13826**	**33.0%**	**59.0%**
MH03	2700	52.3 (1.94%)	19.70	18.64 ± 4.37	**0.0 (0.00%)**	0.07155	**0.04998**	1.53252	**0.12712**	**30.1%**	**91.7%**
MH04	2033	147.5 (7.26%)	18.79	18.35 ± 4.34	**0.0 (0.00%)**	0.07345	**0.06078**	0.50904	**0.20357**	**17.3%**	**60.0%**
MH05	2273	113.2 (4.98%)	19.17	19.29 ± 3.94	**0.0 (0.00%)**	0.10554	**0.05956**	0.25416	**0.15856**	**43.6%**	**37.6%**
V101	2912	236.1 (8.11%)	18.67	18.51 ± 4.37	**0.0 (0.00%)**	0.05487	**0.02746**	0.44819	**0.06578**	**49.9%**	**85.3%**
V102	1710	328.6 (19.22%)	16.77	16.69 ± 4.61	**0.0 (0.00%)**	0.06395	**0.05821**	0.14220	**0.12282**	**9.0%**	**13.6%**
V103	2149	174.9 (8.14%)	18.61	18.96 ± 4.02	**0.0 (0.00%)**	0.10260	**0.04890**	0.84995	**0.12019**	**52.3%**	**85.9%**
V201	2280	207.0 (9.08%)	18.56	14.00 ± 4.80	**0.0 (0.00%)**	0.22014	**0.05833**	5.13288	**0.10164**	**73.5%**	**98.0%**
V202	2348	178.2 (7.59%)	18.78	15.59 ± 4.95	**0.0 (0.00%)**	0.06498	**0.05642**	0.39721	**0.12036**	**13.2%**	**69.7%**
V203	1922	119.3 (6.21%)	18.88	19.46 ± 3.77	**0.0 (0.00%)**	0.07924	**0.06357**	0.33233	**0.17238**	**19.8%**	**48.1%**
Average	–	304.5 (11.5%)	18.10	16.92 ± 4.46	**0.0 (0.00%)**	0.08876	**0.05467**	0.92824	**0.13744**	**30.8%**	**62.1%**

Table 2. Comprehensive comparison of Realtime Baseline and OASIS-enhanced ORBSLAM3 on EuRoC MAV datasets (10 trials each). The table combines dropped frame statistics, adaptive mask sizes, and key accuracy metrics (Mean and Max ATE). Improvement percentages indicate accuracy gains achieved by OASIS compared to the realtime baseline on Jetson hardware. Bold highlights superior values.

Table 3 highlights our method's ability to maintain performance with no computation constraint. While some datasets appear to have large percentage differences, these differences are only a few centimeters. For example, V102 exhibits a drop from 0.05901m to 0.06152m, a difference of 2.5cm, which appears as a -35% difference, but this is insignificant.

In addition, the mask size analysis confirms consistency in our adaptive strategy, with an average mask size of 22, indicating frequent use of the entire frame.

Intel Host Machine (i7-6950X, 64GB RAM)											
Dataset	**Realtime Baseline**			**OASIS**		**Mean ATE (m)**		**Max ATE (m)**		**Improvement (%)**	
	Frames	**Dropped (%)**	**FPS**	**Mask (Mean ± Std)**	**Dropped (%)**	**Realtime**	**OASIS**	**Realtime**	**OASIS**	**Mean ATE**	**Max ATE**
MH01	3682	4.1 (0.11%)	19.98	21.99 ± 0.25	**0.0 (0.00%)**	**0.05775**	0.06016	**0.14362**	0.14650	**-4.2%**	**-2.0%**
MH02	3040	0.8 (0.03%)	19.99	22.00 ± 0.16	**0.0 (0.00%)**	0.04760	**0.04070**	0.13850	**0.11136**	**14.5%**	**19.6%**
MH03	2700	0.5 (0.02%)	20.00	22.00 ± 0.18	**0.0 (0.00%)**	**0.04775**	0.04858	**0.12137**	0.12186	**-1.7%**	**-0.4%**
MH04	2033	0.6 (0.03%)	19.99	22.00 ± 0.25	**0.0 (0.00%)**	0.05831	**0.05647**	0.21220	**0.20627**	**3.2%**	**2.8%**
MH05	2273	1.7 (0.07%)	19.98	21.99 ± 0.26	**0.0 (0.00%)**	**0.06022**	0.06296	**0.15516**	0.16656	**-4.6%**	**-7.3%**
V101	2912	0.0 (0.00%)	20.00	22.00 ± 0.15	0.0 (0.00%)	0.02942	**0.02635**	0.07487	**0.06689**	**10.4%**	**10.7%**
V102	1710	0.0 (0.00%)	20.00	22.00 ± 0.00	0.0 (0.00%)	**0.05901**	0.06152	**0.09623**	0.13031	**-4.3%**	**-35.4%**
V103	2149	0.0 (0.00%)	20.00	22.00 ± 0.00	0.0 (0.00%)	0.04947	**0.04598**	0.12213	**0.10709**	**7.1%**	**12.3%**
V201	2280	0.0 (0.00%)	20.00	22.00 ± 0.00	0.0 (0.00%)	0.06680	**0.06534**	**0.10536**	0.10806	**2.2%**	**-2.6%**
V202	2348	0.0 (0.00%)	20.00	22.00 ± 0.00	0.0 (0.00%)	**0.05461**	0.05616	0.12008	**0.11814**	**-2.8%**	**1.6%**
V203	1922	0.0 (0.00%)	20.00	22.00 ± 0.00	0.0 (0.00%)	0.06940	**0.06778**	0.23502	**0.15970**	**2.3%**	**32.0%**
Average	–	0.7 (0.0%)	20.00	22.00 ± 0.11	**0.0 (0.00%)**	0.05458	**0.05382**	0.13860	**0.13116**	**2.0%**	**2.8%**

Table 3. Comparison of Realtime Baseline (deadlines) and OASIS on EuRoC MAV datasets running on Intel based Host Computer. Each configuration sampled 10 trials each. Bold values denote better performance.

Table 4 compares trajectory accuracy on the Jetson Orin NX across 3 different methods: PID, ω SLAM adaptations, and our proposed *OASIS* method. Each are evaluated alongside two fixed masks (4×4 and 6×6). Mean and maximum absolute translational error (ATE) are reported for the EuRoC MAV sequences.

Results show that OASIS achieves the lowest errors overall when realtime constraints are present: $\overline{\text{ATE}}_{\text{mean}} = 0.054\,\text{m}$ and $\overline{\text{ATE}}_{\text{max}} = 0.137\,\text{m}$.

In contrast, enforcing realtime execution with PID and ω controllers yields larger error errors ($\overline{\text{ATE}}_{\text{max}} = 0.771\,\text{m}$ and 0.688 m, respectively). And although fixed masks produce decent errors, (0.413 m for 4×4, 0.176 m for 6×6), our method demonstrates additional possible reduction in error.

These results indicate that spatially adapting the feature mask is an effective use of limited computation compared to changing frame rate or number of features extracted.

Jetson Orin NX 16GB Developer Kit														
Dataset	**PID**		**Realtime PID**		ω		**Realtime** ω		**Fixed Mask 4x4**		**Fixed Mask 6x6**		**OASIS**	
	Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**
MH01	0.06193	0.15124	**0.05580**	0.22561	0.06121	**0.14756**	0.06893	0.19881	0.08646	0.19617	0.07071	0.17034	0.07001	0.18119
MH02	**0.04124**	**0.10936**	0.05345	0.18994	0.04174	0.11118	0.04654	0.18109	0.07912	0.50470	0.04637	0.14912	0.04817	0.13826
MH03	0.04776	0.12344	0.06769	0.90827	**0.04655**	**0.11926**	0.07157	1.47305	0.06321	0.14696	0.05001	0.12843	0.04998	0.12712
MH04	0.06180	0.21851	0.07906	0.68295	**0.05398**	0.22250	0.06967	0.66827	0.10116	0.36963	0.07072	**0.20209**	0.06078	0.20357
MH05	0.06023	**0.14615**	0.09816	0.26446	0.07123	0.16950	0.09723	0.22950	0.16432	0.39614	0.09982	0.25447	**0.05956**	0.15856
V101	0.02891	0.06886	0.03684	0.36629	0.02676	0.07098	0.03325	0.20681	0.03152	0.07574	**0.02572**	**0.06246**	0.02746	0.06578
V102	0.05920	0.09761	0.05947	0.11463	0.05953	**0.09623**	0.05952	0.11700	0.06105	0.13819	0.06002	0.11030	**0.05821**	0.12282
V103	0.04905	0.12569	0.07280	0.92312	0.04909	**0.11564**	0.11447	0.47894	0.05336	0.20734	0.05116	0.23117	**0.04890**	0.12019
V201	0.06395	0.11781	0.17931	3.68673	0.06643	0.11122	0.14795	2.84008	**0.05513**	0.11418	0.05584	0.11149	0.05833	**0.10164**
V202	**0.05468**	0.11710	0.06879	0.59796	0.05507	**0.11506**	0.06419	0.67893	0.05864	0.28833	0.05683	0.13271	0.05642	0.12036
V203	0.06807	0.49457	0.07344	0.51727	0.07169	0.23764	0.09092	0.50070	0.12834	2.10927	0.09207	0.38178	**0.06357**	**0.17238**
Average	**0.05426**	**0.16094**	**0.07680**	**0.77066**	**0.05484**	**0.13789**	**0.07857**	**0.68847**	**0.08021**	**0.41333**	**0.06175**	**0.17585**	**0.05467**	**0.13744**

Table 4. Mean and maximum absolute translational error (ATE) for the controller variants across EuRoC MAV datasets. Realtime variants refer to enforcing realtime constraints. Lower values are better; bold highlights the best score per dataset regardless of realtime constraints. All measurements are done in meters. Each configuration run with 10 independent trials.

Table 5 shows impact on accuracy when subjected to periodic computational loads with a 10% duty–cycle, 10 second interval load. Without realtime constraints the two platforms perform similarly. The average mean ATE differs by only

0.7 mm and the average maximum ATE by 5 mm, indicating that hardware alone does not bias accuracy when all frames are processed to completion.

The enforcement of realtime deadlines affects the two systems very differently. On Intel, the periodic stress increases the mean ATE by 17% (to 0.0639 m) and the worst-case error by 83 % (to 0.248 m). On Jetson the same deadlines inflate the mean ATE by 115 % (to 0.116 m) and the maximum ATE by an order of magnitude (to 1.336 m), with the largest spikes occurring in sequences V103 and V201.

When OASIS is applied, the performance rebounds, the average mean ATE falls to 0.056 m and the maximum ATE to 0.154 m, reductions of 52 % and 89 %, relative to the realtime baseline with periodic stress. OASIS achieves close to ideal (Data Ready) performance, showing that the per-frame computation budget prediction can adapt to different platforms and unpredictable and uncontrollable interference.

Dataset	**Data Ready with Periodic Stress**				**Realtime with Periodic Stress**				**OASIS with Periodic Stress**			
	Intel		**Jetson**		**Intel**		**Jetson**		**Intel**		**Jetson**	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
MH01	0.05820	0.14950	0.05982	0.15285	0.05839	0.15657	0.05572	0.27804	0.06090	0.16455	0.06592	0.17073
MH02	0.04098	0.09811	0.04248	0.10159	0.04353	0.10591	0.05705	0.78745	0.04444	0.11971	0.04715	0.14499
MH03	0.04785	0.12239	0.04775	0.12155	0.04785	0.12090	0.09187	1.04494	0.04738	0.12229	0.04962	0.12498
MH04	0.05768	0.21354	0.05885	0.21290	0.07674	0.97332	0.09286	0.18171	0.06334	0.20355	0.06726	0.21080
MH05	0.06246	0.15796	0.05848	0.14623	0.05375	0.12361	0.11216	0.31176	0.07311	0.18701	0.06780	0.17571
V101	0.02782	0.06754	0.02761	0.06914	0.02795	0.08503	0.03578	0.10612	0.02673	0.06731	0.02699	0.06554
V102	0.05923	0.10827	0.05881	0.10015	0.05674	0.09492	0.05887	0.11145	0.06000	0.10632	0.05937	0.11455
V103	0.04853	0.11535	0.04953	0.12442	0.15723	0.59109	0.17527	1.43372	0.06002	0.18909	0.05001	0.12309
V201	0.06605	0.10776	0.06483	0.10922	0.05722	0.10928	0.43918	9.90185	0.05707	0.10655	0.05820	0.10533
V202	0.05539	0.11692	0.05520	0.13055	0.05669	0.11944	0.06510	0.21835	0.05630	0.13906	0.05580	0.12957
V203	0.07608	0.23701	0.06911	0.16952	0.06641	0.24901	0.08951	0.32356	0.11471	0.29116	0.07152	0.32726
Average	0.05457	0.13585	0.05386	0.13074	0.06386	0.24810	0.11576	1.33627	0.06036	0.15424	0.05633	0.15387

Table 5. Mean and maximum absolute trajectory error (ATE) for Intel and Jetson under random periodic stress. Each run-type is executed on both systems; lower values are better. Periodic stress of full compute load was produced at a 10% duty cycle over 10 second period. All measurements are in meters.

5.1 Quantitative Analysis

The following figures illustrate key aspects of the OASIS method:

Figure 5 clearly indicates that OASIS reduces sporadic high-error instances, validating the effectiveness of our dynamic runtime approximation strategy, which selectively processes visual data based on realtime computational availability. Due to early dropped frames due to the deadline constraint, the baseline error is significantly higher than OASIS. ORBSLAM3 is effectively unable to remove the error accumulated and is always dealing with stale data due to further dropped frames, resulting in significant mean error. Significant max error is observed; which can lead to critical failure if route planning and navigation assume only mean error for obstacle avoidance.

Figure 6 highlights the key innovation of our approach - the adaptive adjustment of mask sizes. By dynamically modulating the processing load, the system maintains computational efficiency without sacrificing pose estimation accuracy. Due to no frames being dropped in the experimental data, we must use the mask size changes as a proxy to understand the limits of OASIS.

Figure 7 illustrates that while smaller masks lead to improved processing speed, an overly aggressive reduction in mask size can compromise worst-case performance, emphasizing the need for a balanced approach. This hints at a

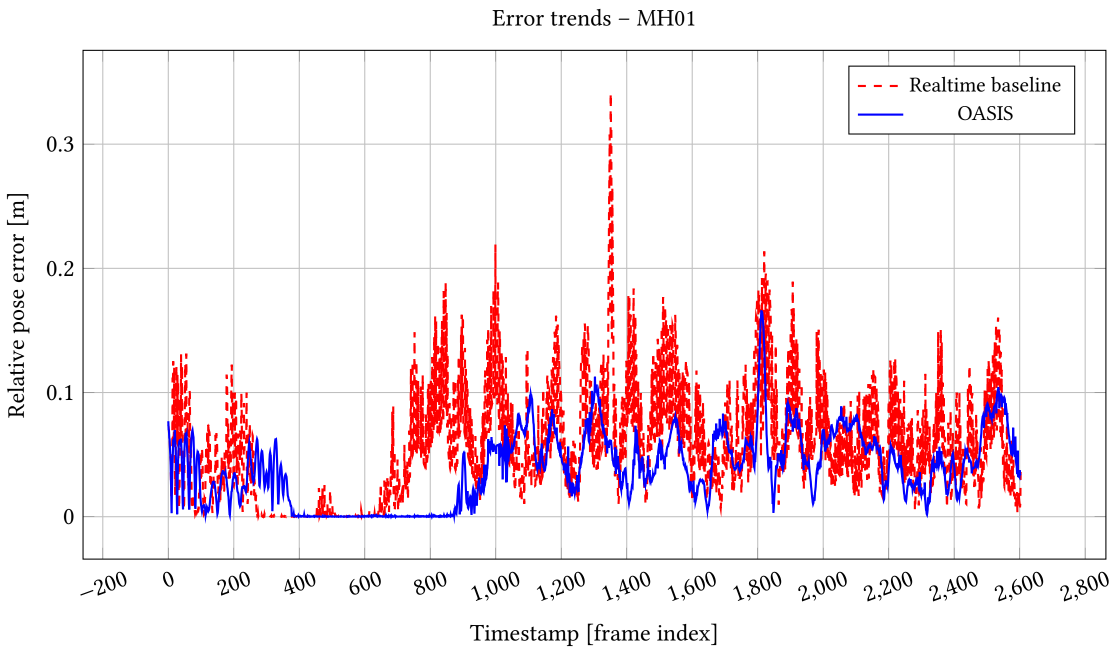


Fig. 5. A representative trial over dataset MH01 to illustrate OASIS impact. The x-axis represents the frame timestamp, and the y-axis shows the relative pose error (RPE) magnitude (in meters). This plot demonstrates that the OASIS method consistently reduces high-error instances encountered in the when deadlines are considered, yielding a more stable and predictable error profile due to enhanced localization from approximated frames.

possible failure mode of OASIS. If the scene forces OASIS to on average select a mask size of $8x8$, a large max ATE will be generated. This behavior appears across the rest of the datasets, see Fig 9 for Machine Hall samples. However, the dataset do not approach this failure point, as shown in Table 2. To better understand this condition, we look at error changes across mask sizes.

Figure 8 confirms that the average tracking performance is stable across a range of mask sizes with a statistical sampling of 10 trials per mask size, reinforcing that the OASIS method is especially effective in mitigating extreme error events rather than altering general behavior or mean behavior of the SLAM. The results generalize across the rest of the datasets (see Fig 10 for results of the Machine Hall Room sets).

5.2 Discussion

The results presented above yield several important insights and highlight the key contributions of this work:

(1) **Dynamic Runtime Approximation:** The OASIS approach operates fully online by dynamically adjusting the mask size based on realtime computational metrics. This enables the system to flexibly allocate processing resources in embedded environments, where computational budgets may fluctuate. The evidence from Fig 6 and Fig 5 supports the effectiveness of this strategy in maintaining robust tracking performance under diverse conditions.

(2) **Significant Reduction in Worst-Case Errors:** The most striking improvement is observed in the reduction of maximum absolute errors (up to 71.9%, as shown in Table 2). This reduction is critical for applications requiring

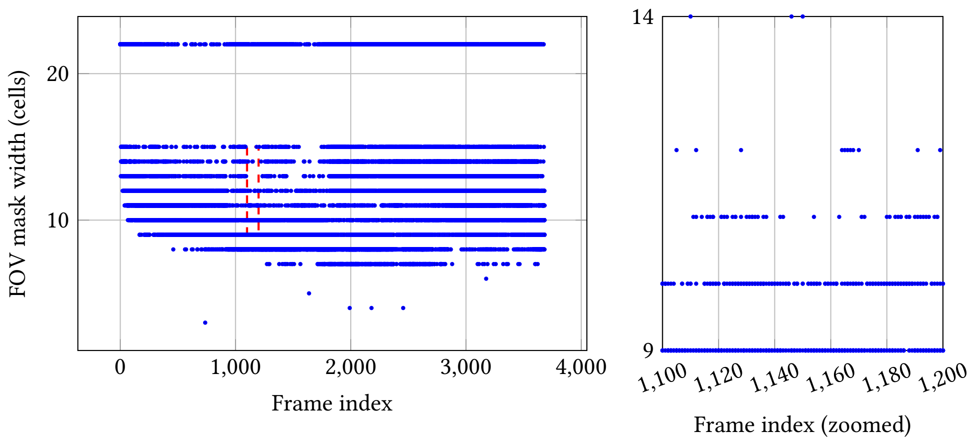


Fig. 6. Left: Adaptive mask sizing chosen by OASIS over a single complete trial to illustrate OASIS masking. The x-axis denotes the frame timestamp of the dataset MH01, while the y-axis represents the dynamically chosen mask size by OASIS. This figure demonstrates how the algorithm leverages realtime tracking performance to predict the computational budget for upcoming frames and optimizing the mask size selection given the budget. Right: Same result as above; zoomed in to show temporal detail between frames. OASIS is switching between mask sizes to deal with scene complexity and a fixed computational budget.

high reliability, as it demonstrates that our method can effectively curtail catastrophic error instances while preserving overall system accuracy.

5.3 Interpretation of Results

The data reveals a clear trade-off between computational efficiency and pose estimation accuracy. The dynamic adjustment of the spatial mask size enables the system to capitalize on available computational resources (Table 4). The dynamic budget estimator handles a trade off of cell processing time and trajectory error suggests that our system can tolerate potential error spikes in real time, seen in Table 5.

The stability of the mean error across various datasets (Table 2) and when removing computation constraints (Table 3) suggests the integration of OASIS does not compromise baseline performance or mean ATE significantly of the SLAM algorithm. And in the presence of computational constraints, OASIS can produce a significant improvement in worst case ATE.

5.4 Limitations and Overhead Analysis

Our approach assumes temporal continuity between consecutive frames; this naturally introduces risks associated with abrupt scene changes. To handle these, we presume that the next frame beyond the scene change is also an abrupt change. This results in a very conservative mask size due to the rolling average's effect on mask selection. This falls back to the core principal of extraction of even some data every frame is better than no data due to deadlines being exceeded. However, this may not be the optimal choice in all scenarios. Though the adaptive logic introduces a modest computational overhead at the end of each frame, our results confirm that gains in error reduction and robustness far outweigh these costs. The time complexity of calculating optimal mask size or application of cell computation is significantly less than a single pass through the tracking process of ORBSLAM3, and any impact is accounted for when budgeting the following frame.

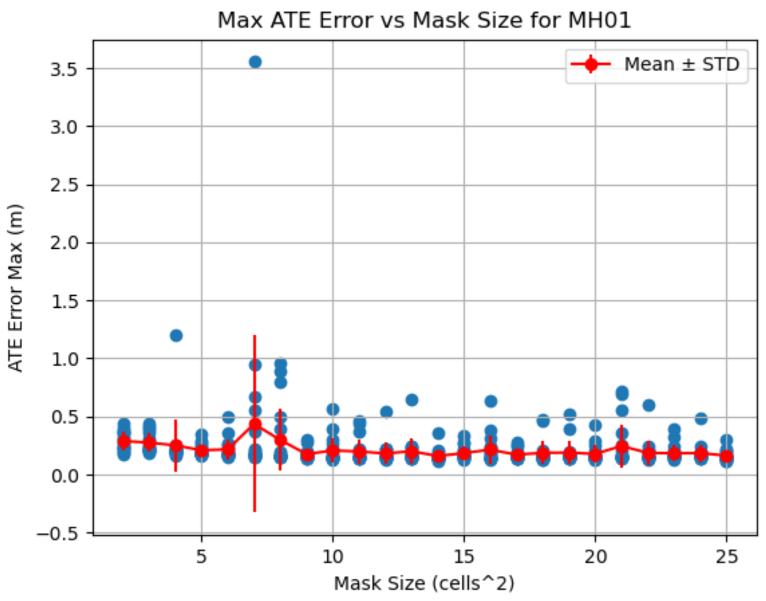


Fig. 7. Effect of mask size on maximum absolute error for dataset MH01 over various mask sizes, 10 trials for each mask size, while enforcing deadlines. The x-axis represents various mask sizes (e.g., 8x8, 16x16), and the y-axis indicates the maximum absolute error (in meters). The plot reveals that reducing the mask size below approximately 8x8 significantly increases the worst-case error, demonstrating a clear trade-off between computational efficiency and accuracy.

6 Future Work and Improvements

The proposed OASIS method shows promising results; however, there are several avenues for future research. One key area is enhancing the approximation method. While our approach currently emphasizes a centralized mask based on the reliable tracking of central points, this assumption may not always hold or may not be optimal. Future work could investigate adaptive strategies that dynamically determine the optimal region for feature extraction on a per-frame basis, possibly incorporating context-aware or learning-based techniques to better identify regions of interest when the center is not the most informative. For example, using velocity as a means to prioritize central cells at high speeds, and edges at low speeds.

Another important direction is extending the method to other SLAM frameworks. Our current implementation requires direct access to the feature extraction code in ORBSLAM3, but adapting OASIS to work with other pipelines could broaden its applicability. This would involve tailoring the dynamic masking and budget estimation processes to different featurizers and SLAM architectures, ensuring compatibility and maintaining performance across various systems. Fig 10 shows that the baseline or mean performance was changed very little; the open question is whether this technique holds for other SLAM implementations.

Real-world application testing and scalability also warrant further investigation. The current evaluation, based on the EuRoC datasets, reflects indoor environments. Future studies should examine how the method performs in outdoor and large-scale scenarios with varying lighting and environmental conditions, assessing its robustness and practical

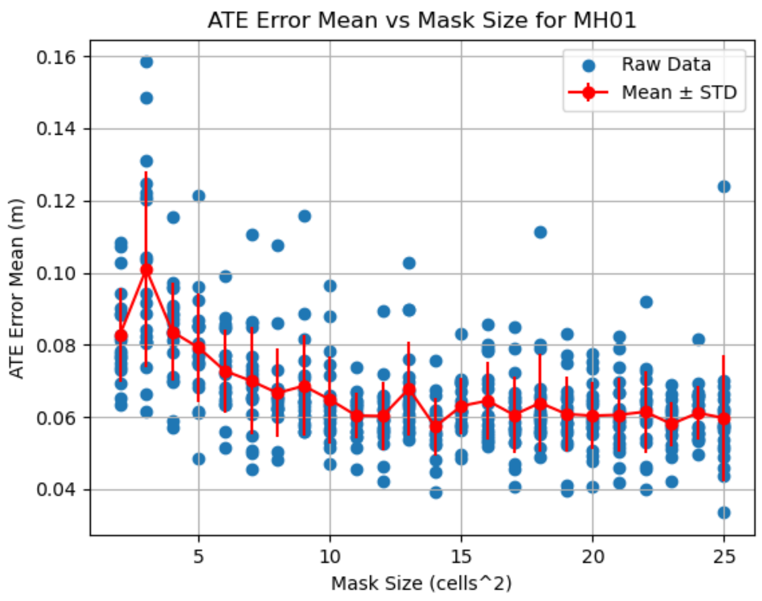


Fig. 8. Impact of mask size on mean absolute error for dataset MH01. The x-axis denotes mask sizes, and the y-axis shows the mean absolute error. The mean error remains relatively robust to variations in mask size, indicating that our adaptive approach predominantly targets outlier error cases without significantly affecting the overall tracking performance.

utility beyond indoor settings. In addition, mask maps (such as Fig 6) may serve as a proxy for segment difficultly, in terms of computation.

To enhance realtime performance, computational offloading through CUDA/GPU or dedicated hardware accelerators should be explored. Offloading computationally intensive operations, such as feature creation and matching, could significantly reduce processing times and enable the handling of higher-resolution images and more complex scenes without compromising system responsiveness.

Finally, an interesting avenue for improvement is the development of a hybrid framework that combines online processing with offline refinement. By integrating a fallback offline module that periodically reprocesses data to refine the map and correct accumulated drift, the system could benefit from the immediate responsiveness of online methods while achieving higher overall accuracy through offline optimization.

7 Conclusion

In this paper, we introduced OASIS, a dynamic adaptive approximation framework specifically developed to enhance realtime responsiveness and accuracy in embedded SLAM systems, particularly ORBSLAM3 deployed on resource-constrained platforms. By adaptively reducing computational load through intelligent masking of input frames, OASIS addresses the significant challenges posed by limited computation resources and stringent timing requirements prevalent in embedded autonomous systems. Our extensive evaluations indicate that adaptive runtime approximations substantially improve worst-case localization performance, highlighting the potential of online adaptive methods in managing the inherent trade-offs between accuracy and computational efficiency. These contributions provide a pathway toward more reliable, predictable, and efficient autonomous navigation for embedded robotics in general.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2348349. The authors thank José Araújo for invaluable feedback on the evaluation, and Jordan Kelley for the excellent brainstorming and naming of the project. We express our gratitude to the CODES+ISSS reviewers for their thoughtful, detailed, and insightful feedback, which greatly enhanced the clarity and quality of this manuscript.

References

[1] Armand Behroozi, Yuxiang Chen, Vlad Fruchter, Lavanya Subramanian, Sriseshan Srikanth, and Scott Mahlke. 2024. SlimSLAM: An Adaptive Runtime for Visual-Inertial Simultaneous Localization and Mapping. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3* (La Jolla, CA, USA) *(ASPLOS '24)*. Association for Computing Machinery, New York, NY, USA, 900–915. doi:10.1145/3620666.3651361

[2] Antoine Billy, Sébastien Pouteau, Pascal Desbarats, Serge Chaumette, and Jean-Philippe Domenger. 2019. Adaptive SLAM with Synthetic Stereo Dataset Generation for Real-time Dense 3D Reconstruction. In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019) - Volume 5: VISAPP*. INSTICC, SciTePress, Prague,Czech Republic, 840–848. doi:10.5220/0007386508400848

[3] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. 2016. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research* 35, 10 (2016), 1157–1163. doi:10.1177/0278364915620033 arXiv:https://doi.org/10.1177/0278364915620033

[4] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. 2021. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. *IEEE Transactions on Robotics* 37, 6 (2021), 1874–1890. doi:10.1109/TRO.2021.3075644

[5] Daniel-André Duecker, A. René Geist, Michael Hengeler, Edwin Kreuzer, Marc-André Pick, Viktor Rausch, and Eugen Solowjow. 2017. Embedded Spherical Localization for Micro Underwater Vehicles Based on Attenuation of Electro-Magnetic Carrier Signals. *Sensors* 17, 5 (2017). doi:10.3390/s17050959

[6] Dorian Galvez-López and Juan D. Tardos. 2012. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics* 28, 5 (2012), 1188–1197. doi:10.1109/TRO.2012.2197158

[7] Tianshuo Guan, Yuchen Shen, Yuankai Wang, Peidong Zhang, Rui Wang, and Fei Yan. 2024. Advancing Forest Plot Surveys: A Comparative Study of Visual vs. LiDAR SLAM Technologies. *Forests* 15, 12 (2024). doi:10.3390/f15122083

[8] Onur Kayiran, Nachiappan Chidambaram Nachiappan, Adwait Jog, Rachata Ausavarungnirun, Mahmut T. Kandemir, Gabriel H. Loh, Onur Mutlu, and Chita R. Das. 2014. Managing GPU Concurrency in Heterogeneous Architectures. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture* (Cambridge, United Kingdom) *(MICRO-47)*. IEEE Computer Society, USA, 114–126. doi:10.1109/MICRO.2014.62

[9] Seongsoo Lee and Sukhan Lee. 2013. Embedded Visual SLAM: Applications for Low-Cost Consumer Robots. *IEEE Robotics & Automation Magazine* 20, 4 (2013), 83–95. doi:10.1109/MRA.2013.2283642

[10] Jiming Long, Fei Wang, Mingyang Liu, Yu Wang, and Qiang Zou. 2025. DOG-SLAM: Enhancing Dynamic Visual SLAM Precision Through GMM-Based Dynamic Object Removal and ORB-Boost. *IEEE Transactions on Instrumentation and Measurement* 74 (2025), 1–11. doi:10.1109/TIM.2025.3547101

[11] D.G. Lowe. 1999. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Vol. 2. IEEE, Kerkyra, Greece, 1150–1157 vol.2. doi:10.1109/ICCV.1999.790410

[12] David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision* 60, 2 (Nov. 2004), 91–110. doi:10.1023/B:VISI.0000029664.99615.94

[13] Filippo Muzzini, Nicola Capodieci, Roberto Cavicchioli, and Benjamin Rouxel. 2023. Brief Announcement: Optimized GPU-accelerated Feature Extraction for ORB-SLAM Systems. In *Proceedings of the 35th ACM Symposium on Parallelism in Algorithms and Architectures* (Orlando, FL, USA) *(SPAA '23)*. Association for Computing Machinery, New York, NY, USA, 299–302. doi:10.1145/3558481.3591310

[14] Vlad Niculescu, Tommaso Polonelli, Michele Magno, and Luca Benini. 2024. NanoSLAM: Enabling Fully Onboard SLAM for Tiny Robots. *IEEE Internet of Things Journal* 11, 8 (2024), 13584–13607. doi:10.1109/JIOT.2023.3339254

[15] Yan Pei, Swarnendu Biswas, Donald S. Fussell, and Keshav Pingali. 2020. A Methodology for Principled Approximation in Visual SLAM. In *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques* (Virtual Event, GA, USA) *(PACT '20)*. Association for Computing Machinery, New York, NY, USA, 373–386. doi:10.1145/3410463.3414636

[16] Julio A. Placed, Jared Strader, Henry Carrillo, Nikolay Atanasov, Vadim Indelman, Luca Carlone, and José A. Castellanos. 2023. A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers. *IEEE Transactions on Robotics* 39, 3 (2023), 1686–1705. doi:10.1109/TRO.2023.3248510

[17] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Las Vegas, NV, USA, 779–788. doi:10.1109/CVPR.2016.91

[18] Yunfan Ren, Fangcheng Zhu, Guozheng Lu, Yixi Cai, Longji Yin, Fanze Kong, Jiarong Lin, Nan Chen, and Fu Zhang. 2025. Safety-assured high-speed navigation for MAVs. *Science Robotics* 10, 98 (2025), eado6187. doi:10.1126/scirobotics.ado6187 arXiv:https://www.science.org/doi/pdf/10.1126/scirobotics.ado6187

[19] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*. IEEE, Barcelona, Spain, 2564–2571. doi:10.1109/ICCV.2011.6126544

[20] Sofiya Semenova, Steven Y. Ko, Yu David Liu, Lukasz Ziarek, and Karthik Dantu. 2022. A Quantitative Analysis of System Bottlenecks in Visual SLAM. In *Proc. of the 23rd Intl. Workshop on Mobile Computing Systems and Applications (HotMobile '22)*. ACM, Tempe, AZ, USA, 7–13. doi:10.1145/3508396.3512882

8 Appendix

Table 6. Default Configuration of ORBSLAM3 or EuRoC MAV Dataset

Parameter	Value	Description
Camera Model	PinHole	Uses the pinhole camera model for projection.
Image Resolution & FPS	752 × 480, 20 Hz, RGB	Defines the image dimensions, frame rate, and color ordering.
Stereo Threshold Depth	60.0	Depth threshold used to distinguish close and far features in stereo matching.
Stereo Transformation	4×4 matrix	Transformation from Camera1 to Camera2 (see YAML for full matrix details).
IMU-to-Camera Transformation	4×4 matrix	Transformation from the IMU (body frame) to Camera1 (see YAML for full matrix details).
IMU Noise Parameters	Gyro: 1.7×10^{-4}, Acc: 2.0×10^{-3}	Noise densities for gyroscope and accelerometer measurements.
IMU Bias Walk	Gyro: 1.9393×10^{-5}, Acc: 3.0×10^{-3}	Random walk (bias diffusion) parameters for the IMU.
IMU Frequency	200 Hz	Sampling rate of the inertial measurement unit.
ORB Features	1200	Number of ORB features extracted per image.
ORB Scale Factor & Levels	Scale Factor: 1.2, Levels: 8	Defines the scale pyramid for multi-scale feature extraction.
FAST Thresholds	Initial: 20, Minimum: 7	Thresholds for the FAST detector to ensure robust corner detection.

accepted 11 August 2025

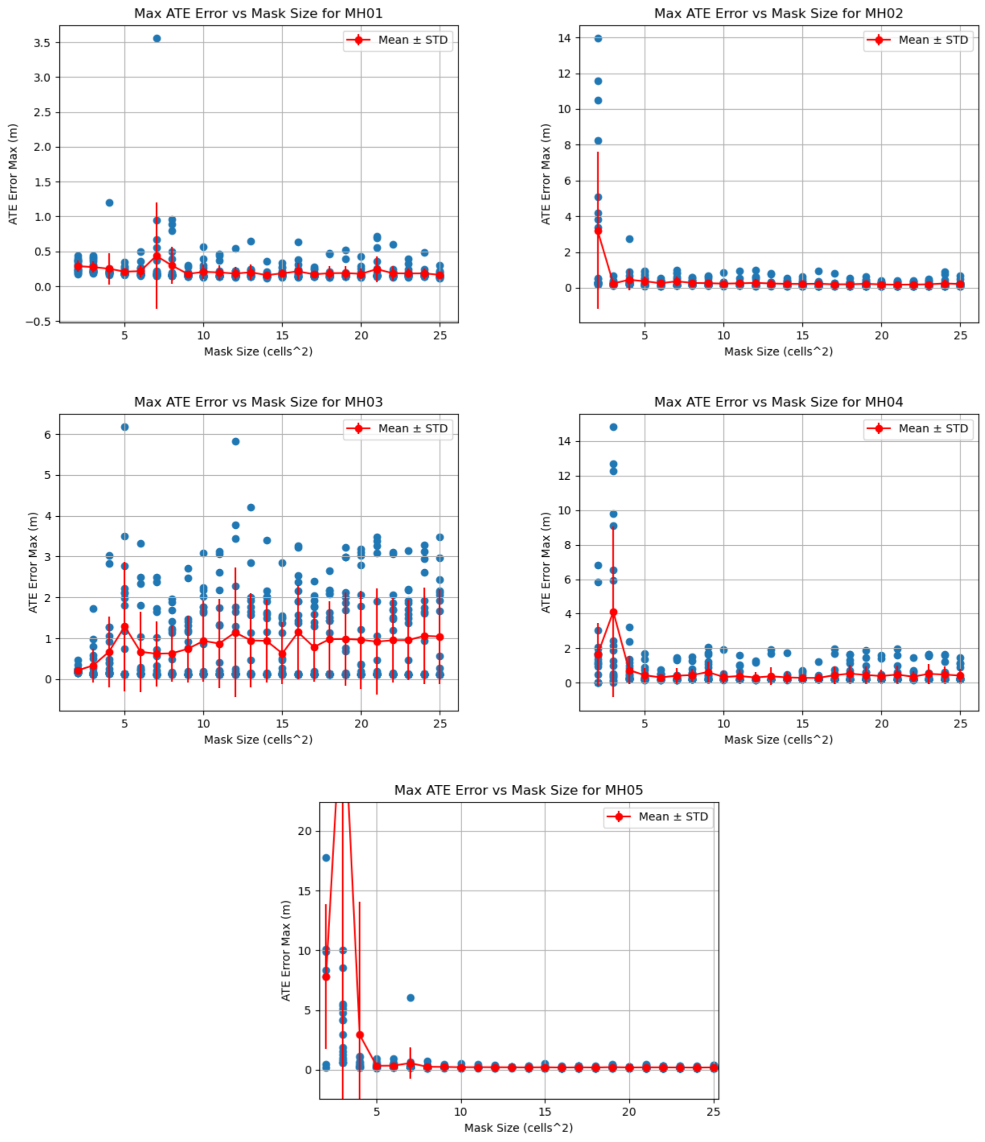


Fig. 9. Change of Max ATE over mask sizes, 10 trials per mask size. Shows similar behavior across all the datasets tested. Shows computational tradeoff of using approximation against worst case trajectory error. Transition from low error to high error serves as the bound on performance improvement with OASIS.

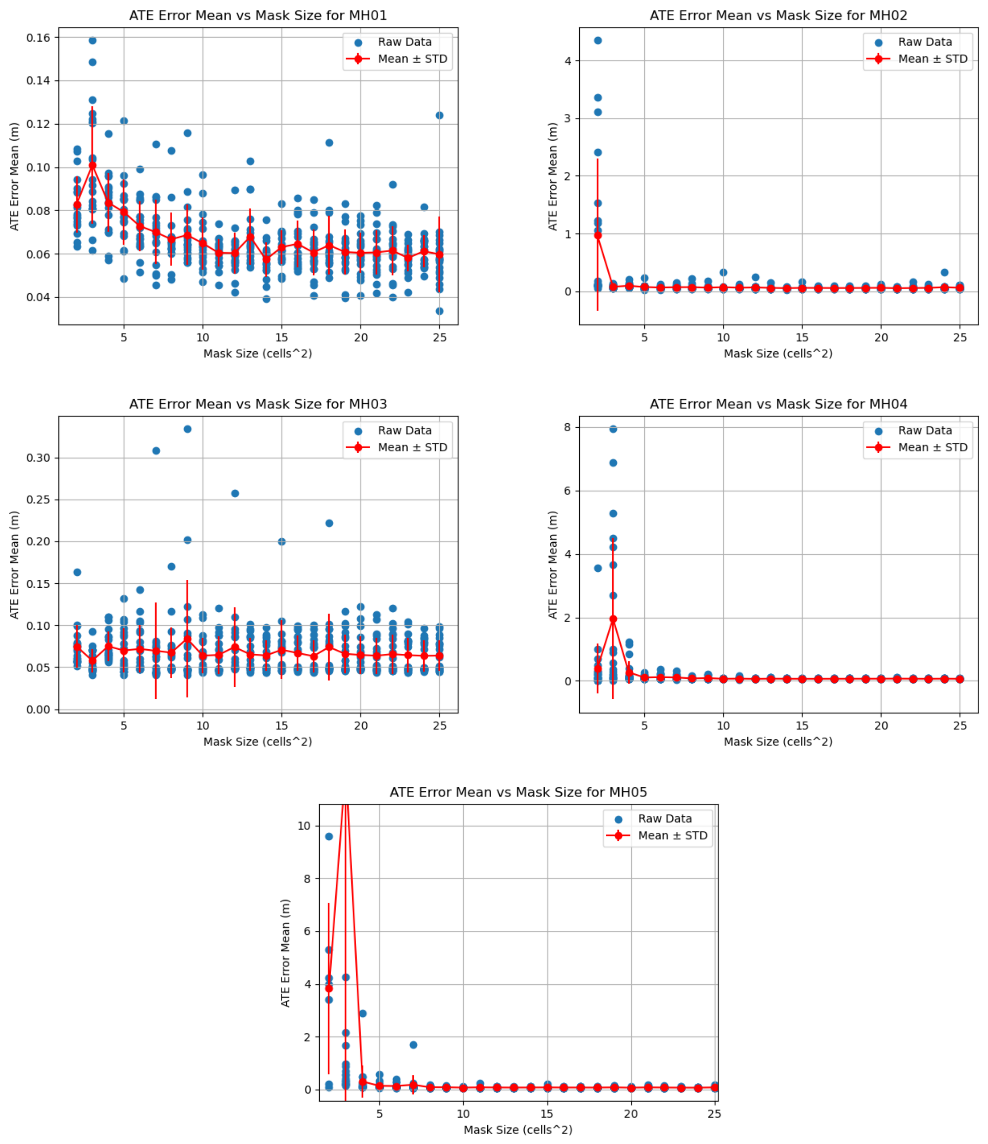


Fig. 10. Change of Mean ATE over mask sizes, 10 trials per mask size. Shows similar behavior across all datasets tested. OASIS generally has little impact to average performance.

Algorithm 1 OASIS: Optimized Adaptive System for Intelligent SLAM

```
1: Initialize:
2:     $T_{\text{frame}} \leftarrow$ allowed time per frame (from SLAM config)
3:     $m_{\text{min}} \leftarrow$ minimum mask size (default to 2)
4:     $L \leftarrow$ number of pyramid levels (from SLAM config)
5: for each new frame $n$ do
6:     $T_{\text{actual}}(n) \leftarrow$ actual frame processing time
7:     $C(n) \leftarrow$ number of cells processed in frame $n$
8:     $cellsProcessed$.append($C(n)$)
9:                                        ▷ Compute simple moving average of cells
10:    $\bar{C}(n) \leftarrow \frac{1}{n} \sum_{i=1}^{n} cellsProcessed[i]$
11:                                       ▷ Compute available budget for next frame
12:    $t_{\text{cell}}(n) \leftarrow \frac{T_{\text{actual}}(n)}{\bar{C}(n)}$
13:    $B_{\text{frame}}(n+1) \leftarrow \frac{T_{\text{frame}}}{t_{\text{cell}}(n)}$
14:    if stereo mode is active then
15:        $B_{\text{frame}}(n+1) \leftarrow \frac{B_{\text{frame}}(n+1)}{2}$
16:    end if
17:                                       ▷ Determine optimal mask size based on budget
18:    $m \leftarrow m_{\text{min}}$
19:    while true do
20:        $C_m \leftarrow 0$
21:        for $l = 0$ to $L - 1$ do
22:            $C_m \leftarrow C_m + \min(N_l, m^2)$
23:        end for
24:        if $C_m > B_{\text{frame}}(n+1)$ then
25:            break
26:        end if
27:        $m^* \leftarrow m$
28:        $m \leftarrow m + 1$
29:    end while
30:    $M(n+1) \leftarrow m^*$                ▷ Optimal mask size for next frame
31: end for
32: Apply Mask During Frame Processing:
33: $FOV_{\text{mask}} \leftarrow$ GenerateMask($M(n+1)$)
34: for all cell in image do
35:     if cell $\in FOV_{\text{mask}}$ then
36:         ProcessCell(cell)
37:     else
38:         SkipCell(cell)
39:     end if
40: end for
```