# WarpSight: Benchmarking SLAM on a GoPro

Michael Gromis, Shri Ishwaryaa S V, Praveen Venkatesh, Soham Bhave
Carnegie Mellon University

***Abstract*—This study introduces WarpSight, a novel modification to Simultaneous Localization and Mapping (SLAM) capabilities for GoPro cameras in extreme environments, with a focus on forest terrains. GoPro cameras, known for their high frame rates and robust design, are explored as versatile sensors for SLAM in GPS-denied environments. We investigate the adaptation of sparse monocular SLAM algorithms, particularly ORB-SLAM2 and ORB-SLAM3, to function efficiently in forest settings. The uniqueness of this research lies in its use of GoPro's high frame rates and inertial measurement unit (IMU) data. We propose an adaptive frame selection network to enhance SLAM efficiency by subsampling frames based on changes in visual and inertial data, aimed at maintaining accuracy while optimizing computational resources. Our experiments, conducted in challenging terrains like Schenley Park, demonstrate the potential of adapting existing SLAM techniques for robust performance in extreme scenarios. Code for our work can be found here: https://github.com/praveenVnktsh/WarpSight-Adaptive-Methods-for-Fast-SLAM-on-GoPro**

## I. Introduction

Portable action cameras, such as the GoPro, have become ubiquitous due to their affordability, robust design, and ease of use. These powerful devices can capture up to 240 frames per second and house an embedded IMU, primarily aiding in picture stabilization. Notably, GoPros are capable of withstanding harsh environments, water, large vibrations, rapid changes in acceleration, and physical impact, making them particularly robust. While their primary use is to capture footage from hobbies and extreme sports, the GoPro proves to be a versatile sensor that can transcend its initial purpose.

Localization and mapping are critical tasks for both humans and machines to perceive and interact with environments. In environments where GPS signals are not available, such as those explored by adventurers and workers like hikers, scuba divers, cave explorers, miners, and tunnel traversers, having a reliable method for mapping and localization becomes crucial to avoid getting lost. The goal of this project is to explore how a GoPro can effectively perform SLAM in extreme environments, with a specific focus on forests.

Various works have been explored to perform monocular SLAM in challenging terrains like forests, as discussed in the next section. Adapting SLAM for GPS-denied environments presents unique challenges and opportunities, and by concentrating on forests, we aim to develop a specialized solution that addresses the intricacies of mapping and localization in these particular settings.

## II. Related Works

Sparse monocular SLAM algorithms are designed to enable a single camera to navigate and map its surroundings in real-time. Unlike dense methods that consider every pixel in an image, sparse SLAM algorithms focus on key features, making them computationally more efficient. These algorithms can be categorized into direct and feature-based methods. Feature-based algorithms involve processes such as feature detection, matching, and optimization, with commonly used feature descriptors like ORB (Oriented FAST and Rotated BRIEF). On the other hand, direct methods take a distinctive approach by optimizing directly for the photometric error between consecutive frames. This approach proves advantageous in scenes with low texture or repetitive patterns. Notable feature-based sparse SLAM algorithms include ORB-SLAM [8] and PTAM [7], while direct methods encompass LSD-SLAM [3] and DSO [2].

Sparse monocular SLAM in forested environments presents numerous challenges. Firstly, GPS data tends to be unreliable under thick canopy cover. Additionally, the forest environment is dynamic, experiencing motion due to wind, substantial changes in lighting conditions, and a high global similarity that could result in significant aliasing. In a comparative study on the performance of state-of-the-art monocular SLAM systems such as ORB-SLAM2 [9], LSD-SLAM, DSO, and SVO [4] on forest data, documented in [5], noteworthy insights emerge. According to the paper, LSD-SLAM and SVO failed to initialize pose tracking on their datasets. In contrast, ORB-SLAM2 and DSO demonstrated reasonable performance in the challenging forest setting. Moreover, there is a growing interest in utilizing SLAM for recovering the spatial structure of forests. Notably, [11] employs a camera and an IMU, similar to our case, indicating a broader trend in leveraging such technologies for understanding and mapping forest environments.

Enhancing the front end of SLAM systems offers various avenues for improvement, and two distinct concepts from the literature involve the incorporation of learning-based methods for feature detection or place recognition, as well as the implementation of adaptive frame rates for visual odometry. In the study referenced in [6], a CNN-based place recognition system known as NetVLAD [1] is integrated with ORB-SLAM2 to enhance loop closure detections. NetVLAD functions as a network designed to generate descriptors, akin to traditional approaches, and undergoes end-to-end training. In [6], despite NetVLAD reliably proposing loop closures to ORB-SLAM2, the SLAM system faces challenges in its frame-to-frame ORB feature matching, highlighting an area with potential for improvement.

A potential enhancement lies in the use of a GoPro's high frame rate, which can significantly boost ORB feature matching performance. Additionally, the adoption of adaptive

keyframe selection, as demonstrated in [10], can potentially reduce computational costs. The subsequent section will delve into the detailed integration of these ideas into our project, with the aim of leveraging the advantages provided by adaptive frame rates for a more robust SLAM front end.

## III. Problem Statement

In this project, we will be exploring the key question of: "How can we optimize a SLAM algorithm for a GoPro to enhance performance and robustness in challenging conditions?"

Using a GoPro, we obtain high frame rate (240FPS) video feeds of challenging environments along with inertial measurements at 200Hz. Since most SLAM techniques are tuned to operate at 30FPS, we propose an adaptive frame selection network that can variably subsample frames from the input video feed based on inertial measurement data and the input camera frames to increase the efficiency of SLAM while maintaining accuracy. When there is relatively little motion, running the algorithm at a full 240FPS can be a waste of resources and can greatly hinder time efficiency on edge systems. Thus, if we are able to selectively choose frames based on information changes in inertial and visual data, we hope to realize efficiency gains in running SLAM algorithms.

Notably, our methodology rests on the assumption that monocular SLAM at 240 FPS performs better than SLAM at 30 FPS because features are better tracked and motion is better estimated between frames. We test this hypothesis in our experiments.

## IV. Methodology

To perform monocular SLAM we employ ORB-SLAM2 and ORB-SLAM3. Because our ORB-SLAM2 implementation is purely monocular, it suffers from scale invariance encountered from purely rotational motions. Over the course of our experiments, we needed to correct for this problem and conducted further tests using ORB-SLAM3. ORB-SLAM3 mitigates the issue of scale invariance by incorporating the GoPro's onboard IMU to perform visual-inertial monocular SLAM.

### A. ORB-SLAM2

ORB-SLAM2 is designed for real-time operation across various environments. It efficiently performs feature extraction, tracking, and map reconstruction using a monocular, stereo, or RGB-D camera setup, making it highly versatile.

*1) Feature Extraction and Matching:* ORB-SLAM2 employs Oriented FAST and Rotated BRIEF (ORB) for efficient feature extraction and matching. This algorithm combines FAST keypoint detector and BRIEF descriptor, tailored for computational efficiency and robustness in diverse environments. KeyPoint Detection is achieved using the FAST algorithm, focusing on distinctive environmental features crucial for accurate matching. Post keypoint detection, the Rotated BRIEF algorithm computes descriptors, ensuring rotation-invariance and noise resistance for stable feature matching.

*2) Frame Tracking:* Continuous camera pose tracking in ORB-SLAM2 matches current frame features with the existing map. This involves motion prediction based on previous frames to provide an initial pose estimate for the current frame, and local feature matching using ORB features between the current frame and the last keyframe to refine the pose estimate.

*3) Local Mapping:* The local mapping module in ORB-SLAM2 updates the map with new information. This comprises keyframe insertion when significant view changes are detected, map point creation by triangulating ORB features matched in multiple keyframes, and local bundle adjustment to optimize keyframes and map points, enhancing local map accuracy.

*4) Loop Closure:* To correct cumulative mapping errors, ORB-SLAM2 features a loop closure detection module. This includes loop detection using a bag-of-words model to identify revisited locations and pose graph optimization upon loop detection to correct accumulated drift, ensuring global map consistency.

### B. Improvements with ORB-SLAM3

Building upon the ORB-SLAM2 framework, ORB-SLAM3 introduces several advanced features to enhance performance, robustness, and versatility, and incorporates the IMU for improved odometry.

*1) Multi-Map System:* ORB-SLAM3 improves on ORB-SLAM2's single-map approach by adopting a multi-map system, allowing for dynamic adaptation in varied environments. It facilitates the creation of new maps in response to significant environmental changes or uncharted areas and ensures efficient handling and transition between multiple maps for continuous operation in complex scenarios.

*2) Visual-Inertial SLAM (VI-SLAM) Integration:* The integration of visual and inertial data in ORB-SLAM3 transforms it into a more robust VI-SLAM system. It combines IMU data with visual inputs to enhance pose estimation and reduce dependency on visual features, offering greater stability in rapid movement scenarios or in environments with inadequate visual data.

*3) Enhanced Robustness and Versatility:* ORB-SLAM3 builds on the robustness of ORB-SLAM2, particularly in challenging environments. It features advanced feature tracking for high-speed movements or low-texture settings and increased versatility across diverse environments and operational conditions.

*4) Atlas Framework:* The introduction of the Atlas framework in ORB-SLAM3 is pivotal for effective multiple map management, crucial in long-term SLAM operations. It enables processing and optimization of individual maps within a unified framework and facilitates efficient long-term operation in dynamically changing environments.

*5) Improved Loop Closure and Relocalization:* ORB-SLAM3 enhances its loop closure and relocalization mechanisms for better map integrity and localization accuracy. This includes more efficient detection of previously visited areas

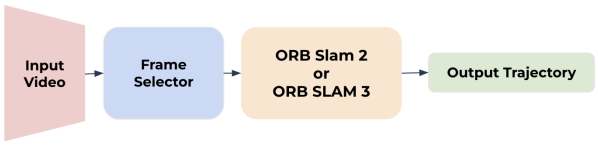


Fig. 1. Adaptive Frame Rate Pipeline

for map correction and enhanced ability for quick recovery from tracking losses, ensuring uninterrupted operation.

*6) Back-end Optimization:* The back-end optimization in ORB-SLAM3 is refined for handling extensive and long-duration mapping tasks. It features improved algorithms for optimizing large-scale maps and camera trajectories, and is optimized for managing extensive maps and prolonged operations while maintaining real-time performance.

### C. Adaptive frame rate

Loosely, we can think of the desired frame rate as being inversely proportional to the relative motion of the camera sensor. Thus, our task becomes akin to estimating the relative motion of the camera within a short time frame, and adaptively choosing frame rate from visual-inertial features. We formulate two distinct methods:

1) Use optical flow-based motion estimation from obtained video feeds and a heuristic function that chooses the frame rate.
2) Use dead-reckoning of IMU data and a heuristic function that chooses the frame rate.

*1) Optical Flow Based Tracking:* To compute the frame rate, for each pair of successive frames, we compute the optical flow between detected Harris corner features in the scene on a grayscale image. This is represented for each feature as a pixel shift.

Since there is a rough correlation between the shift in pixels and the amount of motion in the camera, we use flow as a measure of how much motion the camera undergoes between frames. Since we don't care about the exact motion itself, we use a simple heuristic

$$fps = \frac{240}{((1.0 - \frac{1}{N}\Sigma_{\forall i}\delta x_i)\hat{k} + 1)} \tag{1}$$

where $\delta x_i$ is the feature-wise pixel level shift between adjacent frames, $N$ is the total number of features per frame and $\hat{k}$ is a tunable scaling factor.

Since we are only concerned about varying the frame rate, we empirically choose the parameters of the Harris corner detector such that the features are distributed at a wide range of depths with respect to the camera.

As illustrated in figure 1, once frames sampled by the frame selector module, the frames are fed into the ORB-SLAM2 or ORB-SLAM3, which outputs a map and trajectory.

*2) IMU Based Tracking:* We explored the idea of selecting the frame rate based on the IMU (accelerometer and gyroscope) data obtained from the GoPro. The premise was that higher linear and angular velocities should lead to a higher frame rate.

However, a significant challenge arose due to substantial noise introduced in the data from walking motion without any camera stabilization. Given the difficulty in extracting useful information from the accelerometer data due to the noise, we opted to rely only on gyroscope data to detect turns and dynamically adjust the frame rate during turning instances.

To achieve this, we initially de-noise the gyro data using a Kalman Filter, to filter out noise introduced by walking motion. Subsequently, we calculate the change in rotation (yaw) over a fixed time window and apply heuristics to determine the frame rate, choosing from four options: 30, 60, 120, and 240.

The heuristic categorizes various data ranges into four bins, aligning them with the four available frame rates. The parameters for binning, encompassing the lower and upper limits of each range, are adjustable.

## V. Experiments

### A. Sensor Calibration

The GoPro HERO9 Black required calibration before collecting data. The GoPro provides video footage, 3 axis accelerometer data, and 3 axis gyroscope data, which each requires calibration in addition to Camera-IMU calibration.

TABLE I
Camera Calibration Intrinsic Parameters

| Parameter | Value |
|---|---|
| Focal Length $f_x$ | 201.2 |
| Focal Length $f_y$ | 200.5 |
| Optical Center $c_x$ | 241.3 |
| Optical Center $c_y$ | 136.0 |
| Radial Distortion Coefficient $k_1$ | -0.225 |
| Radial Distortion Coefficient $k_2$ | 0.081 |
| Tangential Distortion Coefficient $p_1$ | 0.00057 |
| Tangential Distortion Coefficient $p_2$ | -0.0181 |

*1) Camera:* The GoPro used with the "wide angle" lens setting, and auto stabilization and auto focusing were disabled.

For the calibration of the camera parameters, the GoPro ROS toolkit was utilized to convert the recorded MP4 files into the Euroc format and ROS bags, suitable for further processing. The camera intrinsics, critical for accurate image analysis and processing, are determined using the Kalibr toolkit, which specializes in multi-camera calibration. Calibration was performed by taking video footage of a checkerboard from multiple angles before feeding into the Kalibr toolkit.

*2) Accelerometer and Gyroscope:* For the Inertial Measurement Unit (IMU) calibration, the Allan Variance ROS package was used. This package is instrumental in obtaining the IMU intrinsics, which include parameters like the noise densities and bias random walk values. These intrinsic values are crucial for the accurate interpretation of IMU data, which plays a significant role in motion tracking and orientation estimation in SLAM processes. The calibrated IMU parameters are vital for the successful integration of inertial data with visual information, especially in a mono-inertial SLAM setup.

*3) Camera-IMU Intrinsics:* Additionally, the camera-IMU extrinsics, which define the spatial relationship between the camera and the IMU, were also calibrated using Kalibr's camera-IMU calibration tool. To integrate these calibration parameters into ORB-SLAM3, a copy of the EuRoC.yaml file is created, and the newly obtained camera and IMU intrinsics, along with the extrinsics, are updated within this file. This step is essential for ensuring that ORB-SLAM3 accurately interprets the sensor data during operation.

### B. Data Collection

We collected one minute of footage in CMU's first floor of the NSH building, completing a full loop of the floor plan. We also collected 19 minutes of hiking footage in Pittsburgh's Schenley Park, revisiting the same places multiple times from different paths. The GoPro was mounted to a teammate's chest with GoPro's "Chesty" chest mount attachment. For footage collected outdoors, we used an iPhone to separately collect GPS data for a ground truth reference. We recognize that GPS data has material measurement noise but found it to be a simple way to get a reliable approximation. Under trees, GPS measurement is likely to exhibit more error; however, most areas in Schenley Park where we collected footage were not overly canopied.

### C. Experiments

1) Monocular SLAM with ORB-SLAM2 on NSH Data: The purpose of this experiment is to validate the hypothesis that higher frames per second (fps) improve localization and tracking in SLAM tasks.
2) Testing ORB-SLAM2 on Schenley Park Data: Here, we further evaluate the performance of ORB-SLAM2 at different frame rates (30 fps and 240 fps) and assess the impact on tracking accuracy and computational time.
3) Testing ORB-SLAM3 on Schenley Park Data: This experiment is designed to determine if ORB-SLAM3, a monocular-inertial algorithm, provides improved results over ORB-SLAM2 in SLAM tasks, particularly in handling scale drift during turns and initialization of inertial tracking at various frame rates.
4) Inertial Tracking Initialization with ORB-SLAM3 Here we investigate the success rate and challenges of initializing inertial tracking in ORB-SLAM3 at different frame rates, especially in scenarios involving significant changes in roll or pitch axes.
5) Inertial and Optical-Flow Based Adaptive Frame Selection Tests Here, we apply our inertial and optical-flow based adaptive frame selection algorithms separately on our ORB-SLAM3 pipeline. We compare their respective performances against GPS and fixed frame rate trajectories.

## VI. Results

This section delves into the results derived from the various experiments outlined earlier. Commencing with the outcomes of monocular tracking using ORB-SLAM2, we then showcase the results obtained through the use of monocular-inertial information with ORB-SLAM3. Finally, we examine the trajectories generated through the adaptive frame rates in conjunction with ORB-SLAM3. Tables of absolute trajectory errors for different trajectories are included in the Appendix.

### A. ORB-SLAM2

The first experiment was performing monocular SLAM using ORB-SLAM2 on the NSH data to validate our hypothesis that higher fps improves localization and tracking. This data consists of a person walking along a square around the seating area in NSH first floor. We tested ORB-SLAM2 on the NSH data at two different fixed frame rates.

We observed that the ORB-SLAM2 loses track of visual features at 30 fps during a turn at a corner. The camera undergoes severe rotation without much translation in this corner. Fig. 2 depicts the lack of features after the sharp turn. However, when ORB-SLAM2 is run at 240 fps on the same data, we find that features are retained and tracking is continued as shown in Fig. 3. Utilizing a higher fps achieves better tracking performance but increases the computation time drastically as more frames must be processed. Another challenge of monocular SLAM is the inability to recover absolute scale due to the lack of depth information. Both the low and high frame rates suffer from this problem.

After verifying our hypothesis, we tested ORB-SLAM2 on 3 minutes of Schenley Park data. The time taken was 45 minutes for 30 fps and 6 hours for 240 fps. Loss of tracking is frequent at low fps causing the trajectory to be very different from the one obtained at high fps as displayed in Fig. 4. The IMU-based adaptive method outputs a trajectory that is closer to the 240 fps trajectory. The number of frames required reduces to 15% of the total frames used at 240 fps. This demonstrates that our adaptive method achieves performance closer to 240 fps with a significantly lower computation time. To improve performance during pure rotations, mitigate scale ambiguity and take advantage of IMU data provided by the GoPro, we shifted to ORB-SLAM3 which is a monocular-inertial algorithm.

### B. ORB-SLAM3

With the expectation of achieving improved results, we tested ORB-SLAM3 on the Schenley Park data. Some key observations summarized from many experiments are discussed here. Firstly, using only monocular data results in scale diminishing or drifting during turns, similar to what was observed for ORB-SLAM2. The sharper the turn, the more pronounced the drift. A comparison of the trajectories obtained at different frame rates using just monocular data to a trajectory given by mono-inertial data can be seen in Fig. 5. It clearly depicts the scale diminishing upon sharp turns for all fixed frame rates. Furthermore, the scale drifts even for gradual turns, as seen in Fig. 6.

The second observation was that we can successfully initialize the mono-inertial tracker only at lower frame rates such as

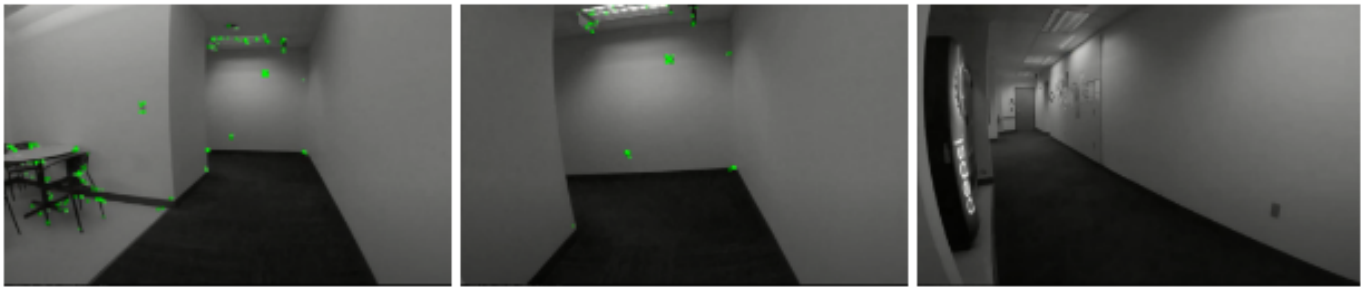

Fig. 2. ORB-SLAM2 on NSH data at 30 fps

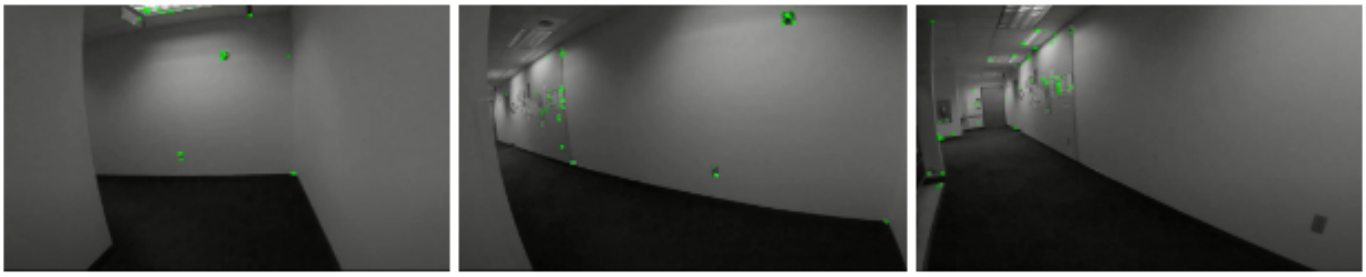

Fig. 3. ORB-SLAM2 on NSH data at 240 fps

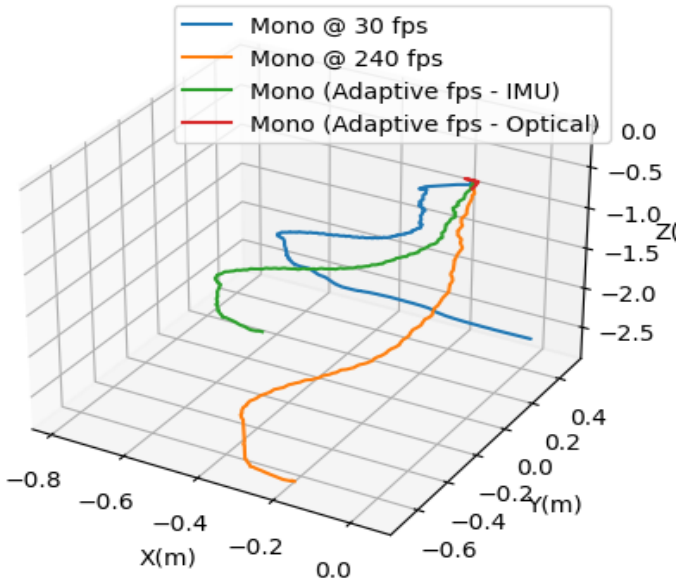


Fig. 4. ORB-SLAM2 on Schenley Park data

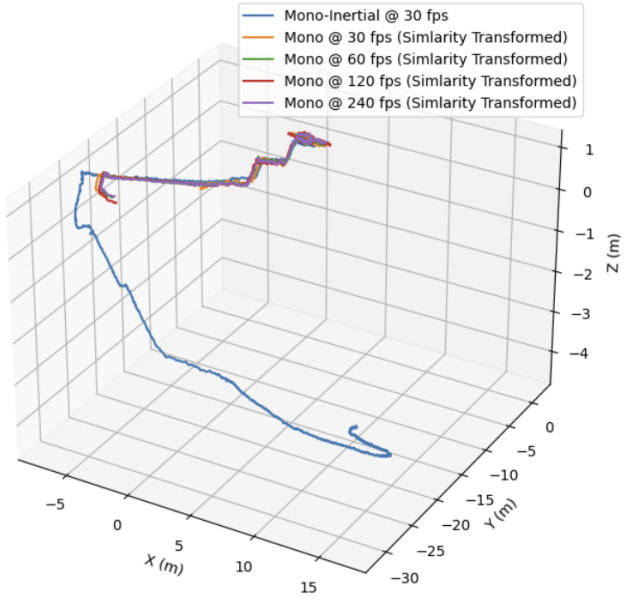


Fig. 5. ORB-SLAM3 on Schenley Park data with sharp turns

30 & 60 fps. At higher frame rates of 120 & 240 fps, there was insufficient change between two consecutive frames, leading to the inertial tracker not initializing on this data. An additional reason is that IMU data is available at 200 fps. This means IMU data is not present between all consecutive frames at 240 fps. A good initialization sequence is crucial for initializing the mono-inertial tracker, involving significant excitation of the roll or pitch axes. Fig. 7 displays trajectories with consistent scale obtained from a good initialization sequence and accurate initialization of inertial odometry.

Even with a well-executed initialization sequence, if inertial tracking is not properly initialized, it may result in scale drifts, particularly during sharp turns. Notably, the success rate for proper initializations is higher at 30 fps compared to 60 fps. The success rate is 4 out of 5 times at 30 fps and 2 out of 5 at 60 fps. The inertial tracking never initializes and causes frequent tracking loss at 120 & 240 fps. If the frame rate is maintained at 30 fps during initialization and switched to 120

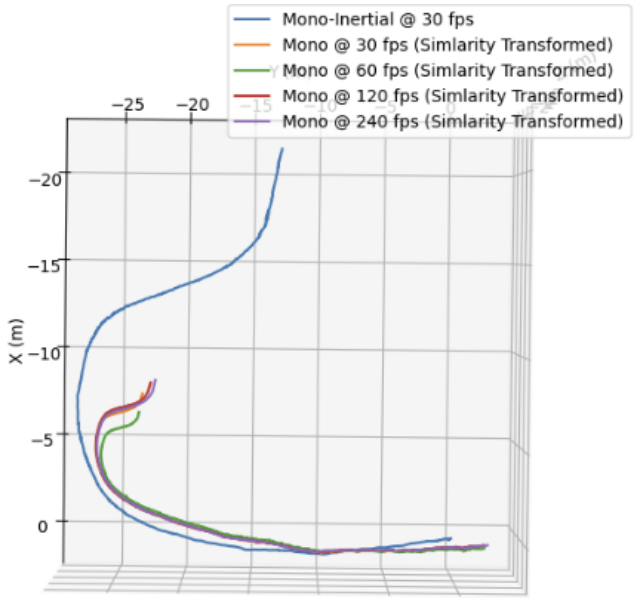


Fig. 6. ORB-SLAM3 on Schenley Park data with gradual turns

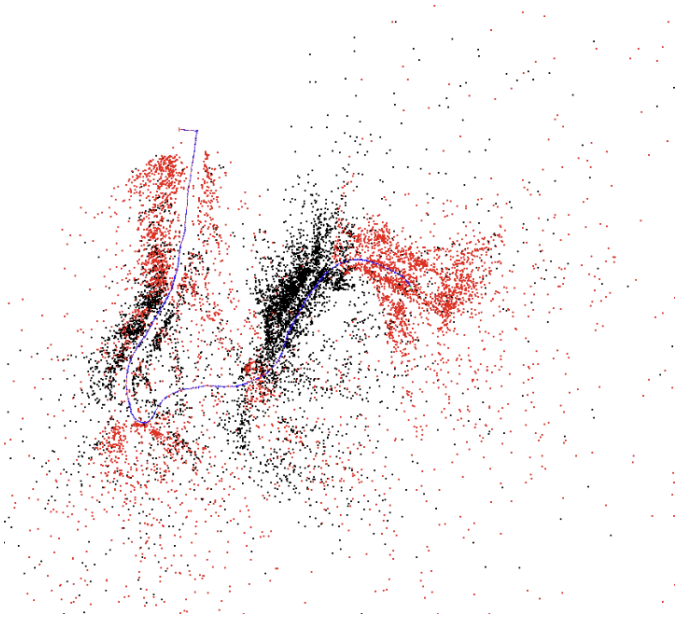

Fig. 8. ORB-SLAM3 map on Schenley Park data

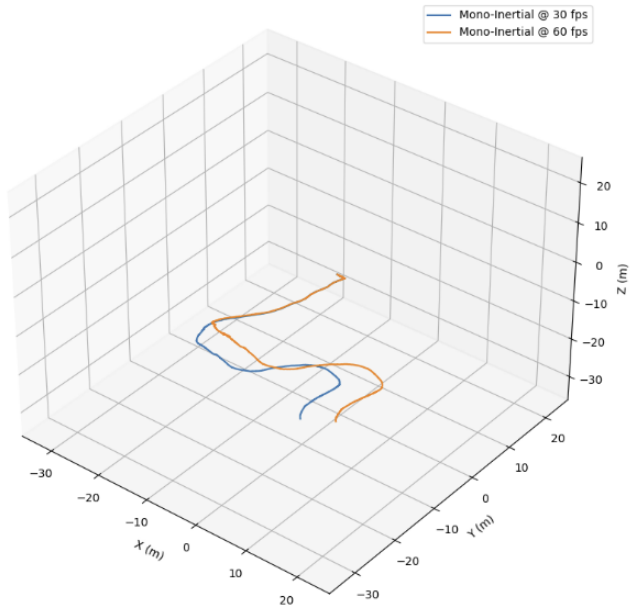


Fig. 7. ORB-SLAM3 on Schenley Park data using mono-intertial information

fps later on, the inertial tracking initializes but is lost once the frame rate changes, after which it is never initialized again.

Finally, the presence of highly similar features in a forest or hiking environment may lead to feature misassociation, as illustrated in Fig. 8. The red points indicate the local map that is being optimized. Points at the beginning of the trajectory are mistakenly included in the local map because features are similar throughout the forest or hiking trail.

### C. Adaptive methods

The two adaptive methods were tested with ORB-SLAM3 using monocular-inertial information. Fig. 9 displays results obtained from the two methods. The IMU-based adaptive method outputted a trajectory that was visually similar to the monocular-inertial trajectory obtained at 30 fps. On the other hand, the optical flow based method's trajectory was closer to the monocular-inertial trajectory obtained at 60 fps.

The average frame rate was around 14 fps for the optical flow method and was 32 fps for the IMU-based method. From our previous experiments, we observed that a higher fixed frame rate provided a better trajectory. Hence, the ability of the optical flow method to output a trajectory closer to the 60 fps trajectory while having a low average frame rate of 14 fps is promising. A significant reduction in computation time is achieved while retaining similar performance.

The deviation of all trajectories from GPS data is noticeable in Fig. 9. We hypothesize the reason for this to be scale diminishing in all the tracked trajectories.

### D. Additional Experiments

In order to compare performance, we collected an extreme video where the camera is moving at high speeds (running at high speed). As a result, the camera is very shaky, and we expect to see a difference in performance. Trajectory comparisons can be seen in fig 10.

From the trajectories and table II, it is clear that the performance of the adaptive techniques in terms of ATE w.r.t 60FPS mono-inertial SLAM is much better than running monocular or mono-inertial at a standard 30FPS, even though the compression ratio is much higher. The monocular optical technique compresses the entire sequence to a mere 5.8% of

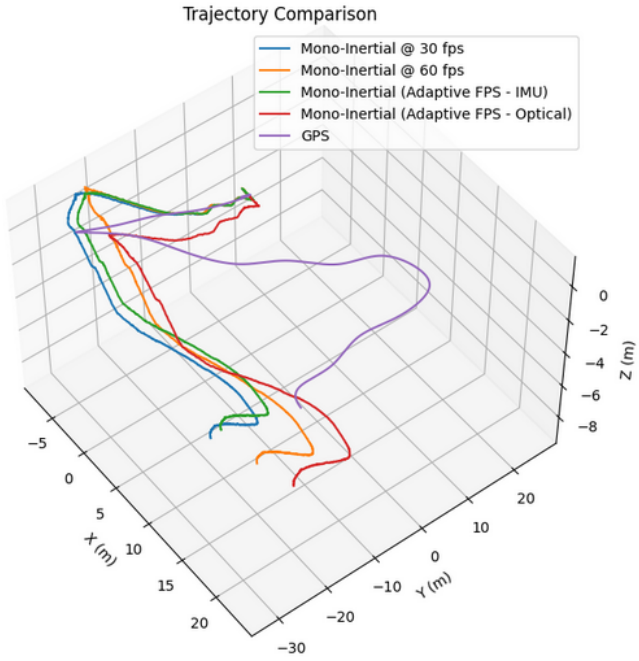


Fig. 9. ORB-SLAM3 on Schenley Park data using adaptive methods

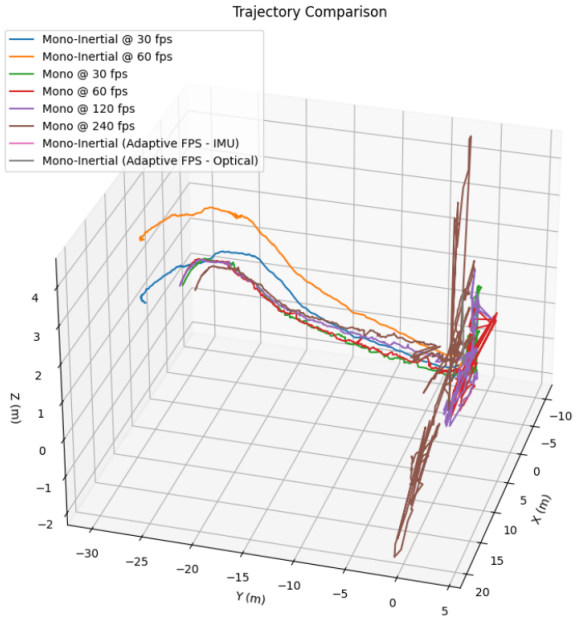


Fig. 10. ORB-SLAM3 on a run

the frames, and yet maintains excellent performance, and beats the best performing monocular method.

We also notice that the errors in the extreme scenario are much higher. This is likely because the 60FPS benchmark is no longer an accurate representation due to the highly dynamic scene.

## VII. Future Work

While our work represents progress in adapting SLAM algorithms for deployment on GoPro cameras in challenging forest environments, several avenues for future research and improvement emerge from our current findings. Further refinement of the adaptive frame selection algorithm is essential to optimize its performance in various forest conditions.

The use of deep learning algorithms for feature association and place recognition can be adopted to mitigate challenges related to feature misassociation. Furthermore, we must investigate effective strategies for initialization of ORB-SLAM3 at high frame rates. Other improvements involve addressing the challenges associated with rapid motion and rotations, ensuring no scale drift in tracking. Finally, the integration of additional sensor modalities provided by GoPro to further enhance SLAM performance can be explored.

## VIII. Conclusion

In summary, this paper presents WarpSight, an innovative approach to adapt SLAM algorithms, specifically ORB-SLAM2 and ORB-SLAM3, for effective deployment on GoPro cameras in forest environments. Our research successfully exploits the high frame rates and robust IMU data from GoPro cameras, addressing the unique challenges of GPS-denied forest terrains. The key contribution of this work is the development of an adaptive frame selection network that intelligently subsamples frames based on changes in visual and inertial data, thereby balancing the trade-off between computational efficiency and SLAM accuracy.

Through extensive experimentation in both controlled indoor environments and the dynamic terrain of Schenley Park, our findings substantiate the hypothesis that higher frame rates enhance SLAM performance, particularly in scenarios involving rapid motion and rotational dynamics, not withstanding the challenges with initialization at a high frame rate. As shown, this improvement in localization and tracking accuracy comes at the cost of increased computational demand.

The integration of visual and inertial data in our adaptive frame rate approach demonstrates significant promise, reducing the required computational load to approximately 15% compared to constant high frame rate processing, while maintaining a trajectory accuracy comparable to that achieved at 240 fps.

Challenges such as feature misassociation in environments with repetitive patterns and the management of IMU noise present avenues for future research. These issues underscore the need for more sophisticated algorithms capable of distinguishing between similar features and effectively filtering sensor noise.

Our work contributes to the field of autonomous navigation in extreme environments by demonstrating the feasibility of employing consumer-grade cameras for robust SLAM applications. Future work will focus on refining the adaptive frame selection algorithm, exploring machine learning approaches for feature discrimination, and enhancing the robustness of SLAM in diverse and challenging environments. This project opens

TABLE II
COMPARISON OF METHODS - TRAJECTORY ERROR (ATE) IS W.R.T 60FPS MONO-INERTIAL SLAM DATA

| Dataset | Technique | FPS | ATE(m) | Retained fraction of frames |
|---|---|---|---|---|
| Schenley Park | Mono-inertial | 30 | 0.06 | 12.5% |
| | | 120 | N/A | 50% |
| | | 240 | N/A | 100% |
| | Monocular | 30 | 0.403 | 12.5% |
| | | 60 | 0.329 | 25% |
| | | 120 | 0.514 | 50% |
| | | 240 | 0.302 | 100% |
| | Monocular Adaptive - IMU | - | 0.313 | 13.3% |
| | Monocular Adaptive - Optical Flow | - | 0.327 | 5.8% |
| Extreme | Mono-inertial | 30 | 4.23 | 12.5% |
| | | 120 | N/A | 50% |
| | | 240 | N/A | 100% |
| | Monocular | 30 | 4.55 | 12.5% |
| | | 60 | 4.79 | 25% |
| | | 120 | 4.91 | 50% |
| | | 240 | 5.96 | 100% |
| | Mono-inertial Adaptive - IMU | - | 5.92 | 20.5% |
| | Mono-inertial Adaptive - Optical Flow | - | 5.67 | 13.6% |

new possibilities for its application in areas like wilderness exploration, rescue operations, and environmental monitoring, where robust and efficient localization and mapping are critical.

## REFERENCES


[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.

[2] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[3] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 834–849, Cham, 2014. Springer International Publishing.

[4] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[5] James Garforth and Barbara Webb. Visual appearance analysis of forest scenes for monocular slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1794–1800, 2019.

[6] James Garforth and Barbara Webb. Lost in the woods? place recognition for navigation in difficult forest environments. *Frontiers in Robotics and AI*, 7, 2020.

[7] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. pages 225–234, 12 2007.

[8] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[9] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

[10] Jin-Chun Piao and Shin-Dug Kim. Real-time visual–inertial slam based on adaptive keyframe selection for mobile ar applications. *IEEE Transactions on Multimedia*, 21(11):2827–2836, 2019.

[11] Fei Yan, Tianshuo Guan, Mohammad Rahmat Ullah, Li Gao, and Yongxiang Fan. A precise forest spatial structure investigation using the slam+ar technology. *Frontiers in Ecology and Evolution*, 11, 2023.


## APPENDIX

The monocular trajectories were scaled to the mono-inertial scale using procrustes analysis.

TABLE III
MONOCULAR @ 30 FPS AS GT OVER FULL PATH

| **Trajectory** | **ATE (m)** |
|---|---|
| Monocular @ 30 fps | 0.959 |
| Monocular @ 60 fps | 0.153 |
| Monocular @ 120 fps | 0.193 |

TABLE IV
MONO-INERTIAL @ 60 FPS AS GT OVER FULL PATH

| **Trajectory** | **ATE (m)** |
|---|---|
| Mono-Inertial @ 30 fps | 4.133 |
| Monocular @ 30 fps | 13.891 |
| Monocular @ 60 fps | 13.578 |
| Monocular @ 120 fps | 13.437 |
| Monocular @ 240 fps | 13.495 |

TABLE V
MONO-INERTIAL @ 60 FPS AS GT OVER PATH BEFORE HITTING THE SHARP TURN

| **Trajectory** | **ATE (m)** |
|---|---|
| Mono-Inertial @ 30 fps | 0.060 |
| Monocular @ 30 fps | 0.403 |
| Monocular @ 60 fps | 0.329 |
| Monocular @ 120 fps | 0.514 |
| Monocular @ 240 fps | 0.302 |

TABLE VI
GPS AS GT OVER FULL PATH

| **Trajectory** | **ATE (m)** |
|---|---|
| Mono-Inertial @ 30 fps | 21.220 |
| Mono-Inertial @ 60 fps | 19.482 |
| Mono-Inertial @ Adaptive fps (IMU) | 20.168 |
| Mono-Inertial @ Adaptive fps (Adaptive) | 18.417 |