iLoco: Real-Time Visual SLAM Using an iPhone

Nikhil Sridhar
niksrid

Velu Manohar
velu

Muhammad Khan
mamankh

Adeep Das
adeepdas

1. Abstract

Simultaneous Localization and Mapping (SLAM) is a foundational component in robotics and augmented reality, enabling systems to understand and navigate unknown environments. While SLAM has traditionally required specialized hardware and intricate calibration procedures, the growing capabilities of consumer devices—particularly smartphones—open the door to more accessible solutions. The iPhone, equipped with LiDAR sensors, cameras, and inertial measurement units (IMUs), provides an ideal platform for developing lightweight and portable SLAM systems.

In this work, we present iLoco, a real-time, plug-and-play visual SLAM system that leverages the iPhone's built-in RGB-D camera and IMU to enable accurate localization. By integrating a sensor suite from the iPhone, iLoco delivers pose estimation utilizing ORB feature matching for RGB-D visual feature extraction and tracking, while GTSAM is employed to tightly integrate inertial measurements with visual odometry for enhanced robustness. The system is engineered to be a "slap-on" solution, requiring minimal setup and no external calibration, making it especially suitable for rapid prototyping, educational demonstrations, and accessible SLAM research. The design of iLoco prioritizes ease of use and adaptability, allowing a wide range of users, from students to developers, to harness the power of real-time SLAM using everyday mobile devices.

2. Related Work

SLAM systems have been developed to address the challenges of accurate and scalable localization and mapping across various platforms and sensing modalities. These systems are designed to provide reliable performance in diverse environments and applications. Many frameworks, such as ORB-SLAM3, integrate multiple sensor types, such as monocular, stereo, and visual-inertial, to enable robust mapping through advanced optimization techniques. Other systems focus on optimizing visual-inertial odometry for mobile and embedded platforms, often utilizing sliding-window approaches for improved efficiency. Some solutions cater to specific use cases, like robotics, by combining sensors such as RGB-D and IMU, though they can be limited by computational constraints and sensor capabilities. Additionally, loop closure techniques, such as Bag-of-Words models, improve map accuracy by enabling spatially consistent feature matching. Our system, iLoco, draws on these ideas, focusing on providing a real-time, accessible solution for localization and mapping on consumer-grade mobile devices with minimal setup or specialized equipment.

2.1. ORB-SLAM3

ORB-SLAM3 is a major advancement in SLAM research, offering a unified, tightly-coupled framework for monocular, stereo, RGB-D, and visual-inertial SLAM. The back-end performs local and global bundle adjustment and pose graph optimization for loop closure. A multi-map architecture allows the system to manage and merge multiple local maps, enhancing robustness in dynamic scenes and long trajectories. Loop closure is handled via a bag-of-words (DBoW2) place recognition system, which detects revisited locations and applies Sim(3) transformations to correct drift. The system also supports re-localization and map reuse, making it suitable for long-term autonomy and AR/VR applications.

In contrast, iLoco is optimized for real-time SLAM on iPhones, leveraging the device's built-in RGB-D camera and IMU without requiring calibration. Data is streamed over TCP and processed using GTSAM, which constructs a factor graph incorporating visual odometry, IMU pre-integration, and loop closure. Although ORB-SLAM3 prioritizes flexibility and precision across various platforms, it requires a specialized setup. iLoco sacrifices some generality for accessibility and ease-of-use, offering a plug-and-play SLAM solution tailored for education, prototyping, and mobile research. [1]

2.2. VINS-Mobile

VINS-Mobile is a state-of-the-art, real-time, tightly-coupled visual-inertial odometry (VIO) system designed for mobile platforms. It extends the capabilities of VINS-Mono to operate efficiently on embedded devices like smartphones and aerial robots. The system fuses monocular camera data

with IMU measurements using a nonlinear optimization-based sliding window approach. In this architecture, a fixed number of recent keyframes are maintained, and their poses are jointly optimized with the associated IMU pre-integrated measurements, resulting in accurate and low-latency motion estimation. To initialize the system, VINS-Mobile includes a robust visual-inertial initializer that estimates scale, gravity, and velocity using visual structure-from-motion followed by IMU alignment.

Compared to iLoco, VINS-Mobile has several similar core components, such as IMU pre-integration, BoW-based loop closure, and sliding window optimization, but differs in key areas. VINS-Mobile is designed for monocular cameras, requiring external mapping tools or dense stereo reconstruction for 3D scene understanding. In contrast, iLoco directly leverages the iPhone's RGB-D camera, simplifying depth acquisition and reducing reliance on structure-from-motion. Furthermore, while VINS-Mobile requires manual sensor calibration and hardware setup, iLoco capitalizes on Apple's built-in sensor fusion and calibration frameworks to offer a plug-and-play experience with zero calibration. Furthermore, iLoco uses GTSAM for back-end optimization, emphasizing modularity and extensibility for educational and prototyping purposes, whereas VINS-Mobile employs a custom, tightly coupled optimization framework optimized for performance on embedded platforms. [2]

2.3. Robot Localization Using Camera & IMU

Attamimi et al. [3] presented a visual-inertial SLAM system for domestic service robots, integrating RGB-D data and IMU measurements from an Intel RealSense D435i. Their implementation used the RTAB-Map algorithm for visual odometry and passed IMU data through a Madgwick filter and an Extended Kalman Filter (EKF) to enhance pose estimation accuracy. The system was deployed on a Jetson Nano and demonstrated the capability to generate both 2D and 3D environmental maps. While effective in environments with rich visual features, their approach faced challenges in low-texture scenes and environments with reflective surfaces, resulting in tracking failures and degraded depth data. Additionally, limitations were observed due to the low IMU refresh rate and computational constraints of the embedded hardware.

Unlike the work of Attamimi et al. [3], our work does not rely on wheel encoders or onboard compute modules, but seeks to use commonly available sensors on an iPhone as the sensing platform, capable of streaming RGB video, depth data using LiDAR, and IMU measurements (accelerometer and gyroscope) over a wireless connection. Through Apple's built-in framework for accessing motion data, Core Motion, we are able to set the IMU refresh rate in Swift code.

2.4. Importance of Loop Closure for Visual SLAM

A variety of methods have been proposed to enhance the robustness and efficiency of loop-closure detection. Although traditional feature-based matching approaches are relatively straightforward, they tend to face challenges related to high computational cost and limited reliability, particularly in the presence of large-scale data, visual ambiguities, and environmental variability.

The work by Zhang et al. (2018) provides critical insights into loop-closure detection using a Bag-of-Words (BoW) model built on ORB (Oriented FAST and Rotated BRIEF) features. Their method effectively tackles the challenge of perceptual ambiguities and cumulative drift common in visual SLAM. By structuring the visual vocabulary into a hierarchical dictionary through k-means++ clustering and employing TF-IDF weighting, their approach achieves significant computational efficiency and robust loop-closure detection. The hierarchical dictionary approach optimizes search efficiency, ensuring fast query responses even as the dictionary increases in size. Furthermore, Zhang et al. improve reliability by integrating spatial consistency checks using Pose Graph optimization to validate detected loop closures, significantly reducing false positives and improving overall system accuracy.

Our iLoco project leverages the robust BoW approach presented by Zhang et al., specifically benefiting from its efficient loop-closure detection mechanism. However, iLoco further advances this approach by incorporating real-time integration of IMU data and RGB-D visual odometry using GTSAM, significantly improving localization accuracy and robustness. Additionally, iLoco differentiates itself by simplifying usability through its design specifically tailored for widely accessible smartphone platforms, eliminating the requirement for external calibration and specialized hardware setups. [4]

3. Method

iLoco is designed to enable plug-and-play real-time SLAM using the built-in iPhone sensor suite. As shown in the diagram below, the system begins by collecting data from the iPhone's LiDAR, RGB camera, and IMU. This multi-modal data is transmitted via TCP to a Linux-based computer that performs data parsing, separating the IMU and visual data streams for subsequent processing. The parsed data is then fed into two parallel pipelines: one for IMU Odometry and the other for Visual Odometry. IMU Odometry estimates motion based on accelerometer and gyroscope data, providing high-frequency pose updates. Simultaneously, visual odometry computes motion estimates by tracking ORB features across RGB-D image frames. These independent pose estimates are then optionally refined through Bag-of-Words, a loop closure module, which

identifies revisited locations using a feature matching approach to reduce drift. All measurements—IMU, visual odometry, and loop closures—are integrated into a factor graph representation using the GTSAM optimization library. The factor graph encapsulates the system's motion and sensor constraints over time, allowing for better trajectory estimation. The final result, as visualized in the bottom-left of the figure, is an accurate trajectory map of the phone's motion through space.

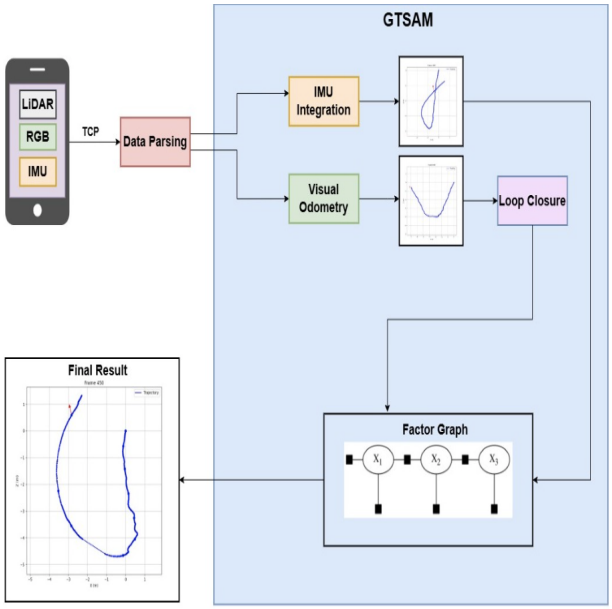

Figure 1. System Architecture for iLoco

3.1. Data Collection

To facilitate real-time sensor data collection for our iLoco project, we developed a Swift application running on a LiDAR-equipped iPhone. The application captures synchronized RGB and depth data using Apple's AVCaptureVideoDataOutput and AVCaptureDepthDataOutput APIs, ensuring tight synchronization through AVCaptureDataOutputSynchronizer. This synchronization aligns each RGB frame precisely with its corresponding depth map from the LiDAR sensor. Additionally, we utilized Apple's CoreMotion framework to collect high-frequency inertial measurements, including accelerometer and gyroscope data, at 100 Hz. These measurements are crucial for sensor fusion, providing essential orientation and motion information that enhances the robustness of our SLAM pipeline.

Our streaming architecture includes two separate TCP streams to optimize data throughput and manage latency effectively. The first stream transmits inertial measurement unit (IMU) data at 100 Hz, while the second stream handles RGB-D data at a frame rate of 30 frames per second (fps). To minimize transmission latency and bandwidth usage, RGB frames are encoded using the H.264 codec directly within our Swift application. On the server side, we decode this H.264 stream in Python, enabling near-instantaneous reconstruction of video frames for real-time processing.

The overarching goal of this design is to achieve real-time SLAM processing capabilities. By carefully balancing sensor data rates, streamlining network communication via dual TCP streams, and employing efficient video encoding and decoding strategies, our system aims to deliver accurate localization and mapping with minimal latency, suitable for dynamic and interactive augmented reality and robotic applications.

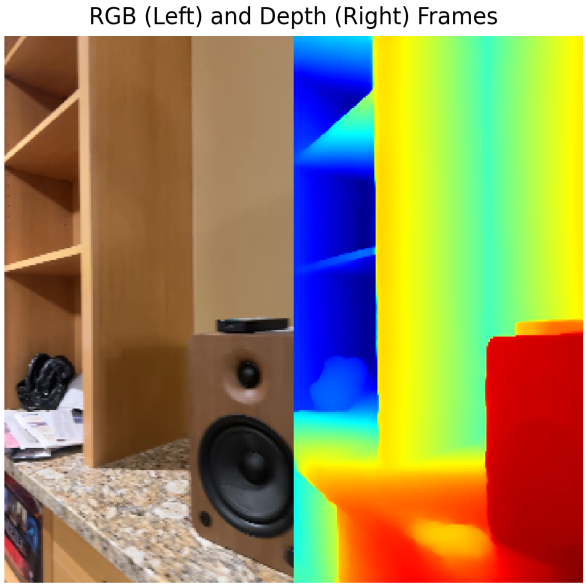

Figure 2. Example of RGB and Depth Data

3.2. IMU Odometry

For IMU odometry, timestamped accelerometer and gyroscope data was collected from the iPhone's IMU, capturing the necessary motion information. The orientation of the data is shown in the figure below. Skew-symmetric matrices were constructed from the angular velocity data, enabling efficient computation of rotational updates. Rodrigues' formula, along with Taylor expansions, was used to handle small-angle approximations for rotation integration.

The IMU state was propagated using discrete-time integration with a zero-order hold assumption, simplifying the process by assuming constant IMU measurements over each time step. With the assumption of piecewise constant acceleration and angular velocity measurements, these values were integrated to update velocity and position states. For rotation updates, the exponential map was applied to the angular velocity vectors, allowing for continuous integration

of angular velocity into a rotation matrix.

As the system moved, velocity and position estimates were updated by integrating accelerometer data, while the rotation matrix was updated using the angular velocity. Bias-corrected IMU data was integrated for a more accurate trajectory representation. By continuously updating these values, the full trajectory of the system, including both positions and orientations, was estimated, providing a real-time odometry solution.

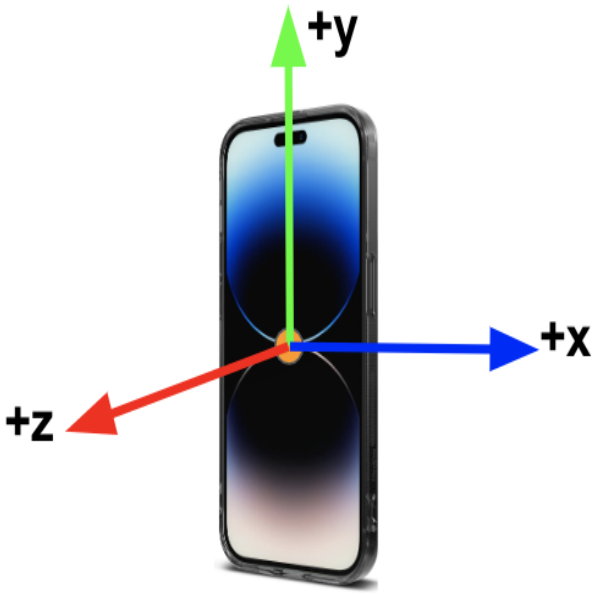

Figure 3. XYZ Orientation for iPhone during IMU Odometry

3.3. Visual Odometry

The visual odometry process began with capturing RGB images from the iPhone's camera and LiDAR depth data. ORB (Oriented FAST and Rotated BRIEF) was used to detect keypoints and extract descriptors from consecutive RGB frames. The keypoints and their corresponding descriptors were then matched using FLANN (Fast Library for Approximate Nearest Neighbors) with LSH (Locality Sensitive Hashing) indexing, which allowed for efficient feature matching across frames. Lowe's ratio test was applied to filter out poor matches, ensuring that only the most reliable correspondences were retained.

Depth maps obtained from the LiDAR sensor were converted into 3D point clouds by applying the camera's intrinsic parameters, which allowed for the projection of 2D keypoints onto the 3D space. From these point clouds, valid 3D points were extracted from the matched 2D keypoints, while invalid depth values were filtered out to improve the accuracy of subsequent calculations.

A 3D rigid transformation between the matched point clouds was then estimated using RANSAC (Random Sample Consensus) to robustly reject outliers. SVD (Singular Value Decomposition) was applied to solve for the optimal transformation that aligned the two point clouds. Finally, the relative transformations between consecutive frames were accumulated, and matrix multiplication was used to compute the absolute poses of the device over time. This allowed for the continuous tracking of the device's trajectory in 3D space.

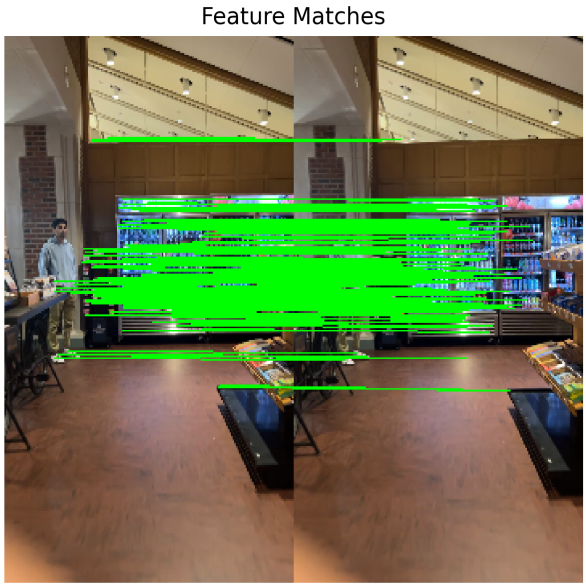

Figure 4. Output from ORB Feature Matching

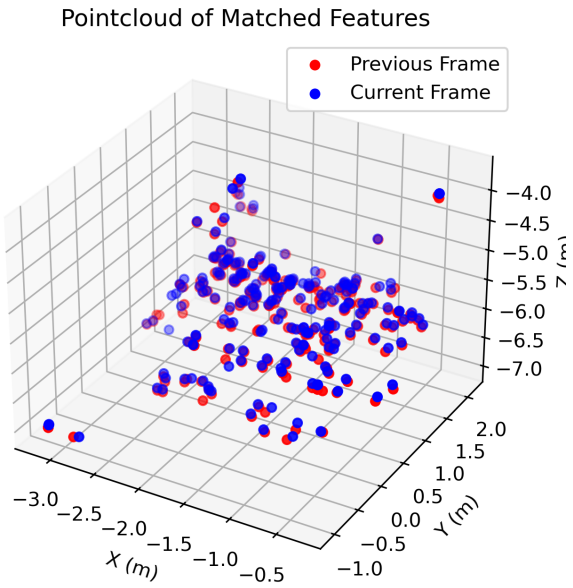

Figure 5. Point Clouds Obtained from Depth Data

3.4. Loop Closure

In our SLAM implementation, we integrated a loop closure mechanism using a Bag-of-Words (BoW) approach combined with ORB features for visual recognition. The purpose of loop closure detection is to identify previously visited locations by comparing visual features across different frames, significantly reducing drift and enhancing map accuracy.

We began by training a visual vocabulary based on ORB descriptors extracted from selected video frames at regular intervals. This vocabulary serves as a reference dictionary, facilitating rapid comparison between images. For each incoming video frame, we extracted ORB descriptors and computed their corresponding BoW representation, which succinctly captures the visual content.

To efficiently detect potential loop closures, we implemented a KD-Tree to store and query the BoW vectors from past keyframes. Upon receiving a new frame, its BoW descriptor was compared against the database using cosine similarity, enabling fast retrieval of visually similar frames. Frames meeting a predefined similarity threshold and temporal spacing criteria were marked as loop closure candidates.

Subsequently, to validate each loop closure candidate, we performed detailed feature matching between the current and candidate frames. Loop closures were accepted only when a sufficient number of feature matches exceeded a certain confidence level, ensuring accurate loop closure detection.

Identified loop closures were integrated into the GTSAM optimization framework, further refining the SLAM trajectory by minimizing accumulated localization errors. However, the loop closure was not as effective as it could have been due to the visual odometry and IMU data being too noisy, and we didn't allow the data to run long enough to create enough constraints to dramatically increase the accuracy of the map.

3.5. GTSAM

We employ GTSAM to perform batch visual-inertial SLAM, combining IMU preintegration with visual odometry to estimate a globally consistent trajectory. Our system builds a factor graph where IMU measurements are integrated between visual keyframes, producing high-frequency motion constraints that capture the system's dynamics. Visual odometry is used to provide relative pose constraints between keyframes, anchoring the solution to observed motion in the environment. The optimization is carried out incrementally using ISAM2, allowing us to refine the trajectory in minibatches without recomputing the full solution from scratch.

This framework is particularly well-suited for our dataset, which includes both high-rate inertial data and sparse RGBD visual observations. By leveraging IMU preintegration and incorporating both motion priors and sensor noise models, we can fuse multi-rate, multi-modal data streams into a single consistent trajectory estimate. The result is a smooth and drift-reduced pose sequence that respects both the local high-frequency inertial dynamics and the global geometric constraints provided by visual odometry. This serves as the backbone for evaluating our downstream video plan prediction and alignment tasks.

4. Results

We evaluate our visual-inertial SLAM system on a dataset of 8 trajectories collected in indoor environments where depth sensing is viable. These trajectories span a variety of common indoor scenes—including kitchens, convenience stores, living rooms, and basements—with each run lasting no more than one minute. During data collection, we enforce a consistent ground-truth trajectory by replaying pre-defined motion paths, allowing for direct comparison between the estimated and true poses.

In Figure 6, we qualitatively compare the final optimized trajectory produced by our graph-based SLAM pipeline to two baseline methods: pure IMU odometry and pure visual odometry. The IMU-only baseline exhibits significant drift over time, as small integration errors accumulate without any correction from external observations. On the other hand, visual odometry alone provides frequent relative pose updates but suffers from jitter and noise due to local feature tracking ambiguities and occlusions. In contrast, our combined method fuses both sources of information using factor graph optimization, resulting in trajectories that are much smoother and closely aligned with ground truth.

We also present a quantitative comparison in Figure 7, where we plot the average pose error as a function of elapsed time across all 8 trajectories. At each timestep, we compute the average translational error as the Euclidean distance between the estimated and ground truth positions, and the average rotational error as the absolute angular difference in the XZ-plane (yaw) between estimated and ground truth orientations. These errors are averaged across all trajectories to obtain smooth trends over time. As expected, IMU-only error grows rapidly with time, while visual odometry maintains a bounded but noisy error profile. Our fused SLAM system consistently achieves the lowest error across all time segments, confirming that the integration of IMU and visual cues through graph optimization substantially improves both short-term accuracy and long-term consistency. These results validate the robustness of our approach in diverse indoor settings.

You can also find our project video at this link: `https://youtu.be/goo5J6C9n0c`

Time (s)	IMU Odometry (Avg ± SD) [m^2]	Visual Odometry (Avg ± SD) [m^2]	Our Method (Avg ± SD) [m^2]
10	0.405 ± 0.075	0.281 ± 0.018	0.292 ± 0.732
20	1.632 ± 0.075	0.541 ± 0.243	0.56 ± 0.332
30	4.542 ± 0.195	1.27 ± 2.549	1.168 ± 1.352
40	8.988 ± 0.465	3.485 ± 1.162	2.14 ± 1.192
50	13.824 ± 1.485	6.13 ± 13.503	3.296 ± 2.088
60	18.6 ± 2.985	7.854 ± 6.778	4.024 ± 2.136

Figure 6. Quantitative Comparison of Algorithms

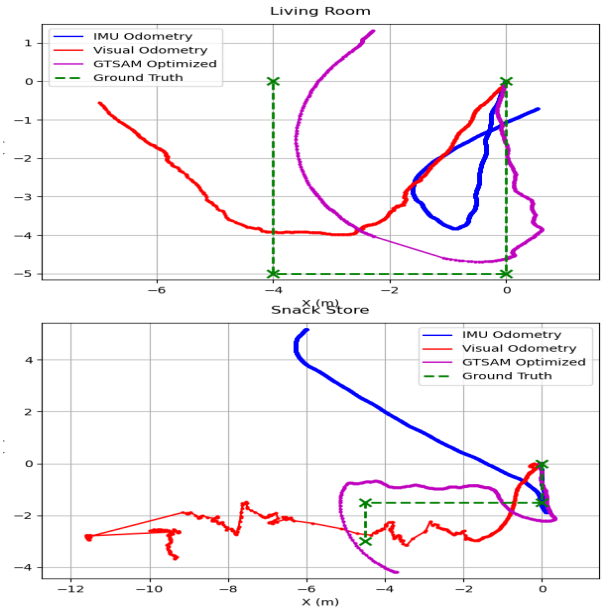

Figure 7. Trajectories from 2 Test Runs using iLoco

5. Conclusion

Our experiments highlight several important performance trends across different localization methods. While IMU-based tracking provides relatively reliable estimates over longer linear motions (greater than 5 meters), it suffers from cumulative drift without external correction. Visual odometry, in contrast, performs reasonably well in textured areas but degrades significantly during sharp turns or in environments with textureless walls, such as hallways or corners. Our GTSAM-based fusion framework offers a balanced solution, with improved robustness and accuracy across diverse trajectories. Notably, using mini-batch optimization enhances convergence stability without requiring full batch processing. Although our fused method (iLoco) consistently outperforms standalone IMU or VO, it still trails behind ARKit in terms of rotational accuracy—especially during dynamic motions or in low-light settings.

Looking ahead, we plan to incorporate full mapping capabilities to support loop closure and improve long-term consistency across repeated traversals. Additionally, we aim to decouple and integrate visual and depth sensing (e.g., LiDAR or structured light) to better handle occlusions, dynamic lighting, and structural sparsity. These improvements will allow our system to generalize more robustly across both structured indoor scenes and more challenging environments.

To support reproducibility and encourage further research, we have open-sourced our implementation. The full codebase is available at: `https://github.com/adeepdas/iLoco`

References

[1] Antoni Rosinol, Ignacio Garcia-Fernandez, Francisco Montagud, Rafael Rivas, and Andrew J. Davison. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM. *arXiv preprint arXiv:2007.11898*, 2020. `https://arxiv.org/abs/2007.11898`

[2] HKUST Aerial Robotics Group. VINS-Mobile: Monocular Visual-Inertial State Estimator on Mobile Phones. 2021. `https://github.com/HKUST-Aerial-Robotics/VINS-Mobile`

[3] Muhammad Attamimi, Paltzky Ainurrafi Hidayat, Hendra Kusuma, and Tasripan. Room Mapping and Robot Localization Using RGB-D Camera and IMU Sensor. In *Proceedings of the 2021 International Conference on Computer, Control, Informatics and Its Applications (IC3INA)*, pages 182–186, 2021. ACM. `https://dl.acm.org/doi/10.1145/3489088.3489128`

[4] Jiachen Zhang, Dahang Ai, Yi Xiang, Xin Chang, Yi Wang, and Xiaodong Chen. Bag-of-words based loop-closure detection in visual SLAM. In *Proceedings of SPIE 10816, Advanced Optical Imaging Technologies*, page 1081618, 2018. 10.1117/12.2500819. `https://doi.org/10.1117/12.2500819`