Front-End and Back-End Optimization of ORB-SLAM3 and VINS-Fusion

Tianyi Cao, Jackson Chen, Conghao Jin, Junhao TU

***Abstract*— This project aims to enhance two established SLAM systems by reorganizing and refining their Front-end and Back-end parts. The modifications are designed to improve image preprocessing and use different optimization methods.**

I. Introduction

Simultaneous Localization and Mapping (SLAM) is essential for robotics and autonomous vehicle navigation, and plays a critical role in generating the map of an unknown environment while keeping track of the objects' pose simultaneously.

In this project, two widely-used SLAM systems, ORB-SLAM3 and VINS-Fusion are re-organized and optimized based on their structures. Specifically, the modification focuses on the front-end and back-end structure of each method. At the end, the results of various optimization algorithms are compared with its original implementation, following its optimization's performance analyses based on EuRoC dataset experiments.

II. Background

A. ORB-SLAM

The front-end of ORB-SLAM3 adopted method of feature points to implement visual odometry [1]. Specifically, it utilizes the Oriented FAST algorithm to detect keypoints and employs the Rotated BRIEF algorithm to compute descriptors for each keypoint.

As the improvements have been conducted to improve the quality and efficiency of the keypoint, thus Oriented FAST algorithm need to be understood. Oriented FAST, an enhancement of the conventional FAST algorithm [2], is engineered to identify areas with significant local pixel intensity variations and assigns a direction to these points. The core principle is that a pixel exhibiting substantial intensity differences compared to its neighbors is more likely to represent a corner [3]. In practice, the algorithm evaluates the brightness of pixels on a surrounding circle for each pixel in the frame. As shown in Fig. 1, a pixel is designated as a keypoint if at least 12 out of the 16 surrounding pixels show intensities either above or below a threshold compared to the central pixel. After detection, the non-maximal suppression technique [4] is employed to ensure only high-quality corners are selected, which helps in preventing the clustering of keypoints. Unlike the standard FAST, Oriented FAST includes enhancements for scale and rotation invariance. Scale invariance is achieved by creating an image pyramid (Fig. 5) and detecting corners at each level, while feature rotation is determined by the intensity centroid. The intensity centroid can be obtained as: In a small image block B, define the moment of the image block as:

$$m_{pq} = \sum_{x,y \in B} x^p y^q I(x,y), \quad p,q = \{0,1\} \tag{1}$$

The centroid of the image block can be found by the moment:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right). \tag{2}$$

Connect the geometric center O and the center of mass C of the image patch to obtain a direction vector $\overrightarrow{OC}$. Therefore, the orientation of feature point is defined as:

$$\theta = \arctan\left(m_{01}, m_{10}\right). \tag{3}$$

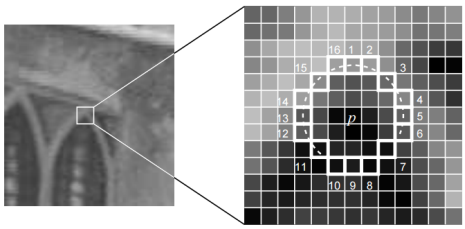

Fig. 1. FAST Featured Points [3]

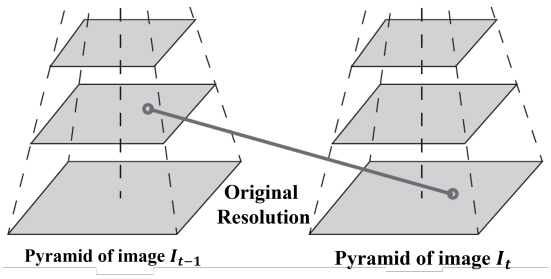

Fig. 2. Use pyramids to match graphs with different zoom ratios

https://github.com/junhaotu2000/ROB-530-Final-Project

The back end of ORB-SLAM3 firstly includes a sophisticated Local Mapping module for building accurate and reliable maps from keyframes generated by the tracking process [1]. This module performs several critical functions: inserting new keyframes into the map, applying Local Bundle Adjustment (BA) to optimize the structure and viewing parameters, and triangulating new map points from unmatched features found in current keyframes. Local Mapping operates by enhancing the map's precision and reducing redundancy through a systematic filtering of map points and keyframes. What's next is doing Loop Closing, a function aims to to mitigate the cumulative error from the front-end odometry. This is achieved by detecting if the robot has returned to a previously visited location and establishing loop constraints within the co-visibility graph. Subsequently, a global BA optimizes the entire map, distributing the large accumulated errors among all the keyframes involved in the loop.

B. VINS-Fusion

The front-end of VINS-Fusion is responsible for collecting critical data utilizing mono/stereo cameras along with an inertia measurement unit [5]. We looked into the method for pre-process of the visual data and understood how VINS-Fusion implements its visual odometry (VIO) on its front-end. Similar to ORB-SLAM, VINS-Fusion also leverages feature-tracker for feature points detection. However, it specifically uses Lucas-Kanade [6] optical flow for multiple tasks, including back & forth tracking the feature points between two frames and left & right tracking between the stereo cameras if they exit.

To be more specific, this sparse optical flow method assumes that the gray-scale of a pixel is a function of position and time, i.e., $\boldsymbol{I}(x, y, t)$. Also, to implement this method in real world, there must be a strong assumption that the gray-scale of a certain pixel does not change with time nor position. From this assumption, we have

$$\boldsymbol{I}(x + dx, y + dy, t + dt) = \boldsymbol{I}(x, y, t) \tag{4}$$

Apply Taylor Expansion on the left hand side,

$$L.H.S. \approx \boldsymbol{I}(x, y, t) + \frac{\partial \boldsymbol{I}}{\partial x}dx + \frac{\partial \boldsymbol{I}}{\partial y}dy + \frac{\partial \boldsymbol{I}}{\partial t}dt \tag{5}$$

With the assumption, we have

$$\boldsymbol{I}(x, y, t) + \frac{\partial \boldsymbol{I}}{\partial x}dx + \frac{\partial \boldsymbol{I}}{\partial y}dy + \frac{\partial \boldsymbol{I}}{\partial t}dt = 0 \tag{6}$$

Now move the t term to the right hand side and divide both side by dt, write it in matrix format, we have

$$\begin{bmatrix} \boldsymbol{I}_x & \boldsymbol{I}_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\boldsymbol{I}_t \tag{7}$$

Now, we make another assumption that the pixels within the same window (of size $w \times w$) have the same displacement w.r.t. the last frame. Thus, we can have w^2 number of equations, which lead the solution of u, and v to a least square root of the set of equations. Note that in VINS-Fusion, $w = 21$.

However, in real-case implementation of this optimization problem, we need to make sure that the calculation is after all converged, which requires the initial value to be close enough to the optimal one. To solve this problem, we introduce multi-level image pyramid in Fig. 3.

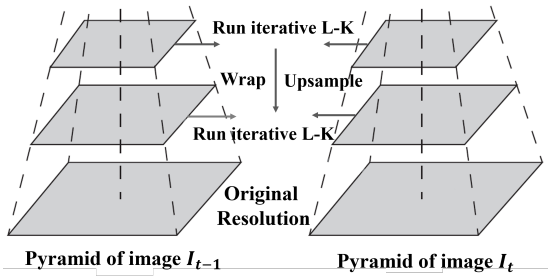

Fig. 3. Image Pyramid and Coarse-to-fine Process [7]

The image at higher level of the pyramid is essentially a blurred and subsampled version of the previous level. Thus, this method is able to avoid the possible divergence when there is a large displacement between two frames. The lower resolution of it reduces the magnitude of the displacement in pixel and thus ensure the convergence of the optimization process.

Such a image-processing method can rapidly process a large number of feature points matching but it requires better quality in the continuity and stability of frames.

The back-end of a VINS-Fusion system plays a critical role in achieving high-precision localization and navigation. It receives pre-processed data from the front-end, which collects and initially processes input from visual and inertial sensors, including data such as feature points, normalized coordinates and IMU data. This back-end integrates the initialization process of camera and IMU, which calibrate the transformation matrix, scale factor and relative parameters regarding the sensor correlation. Basing on this calibrated data, the pose and world coordinates can be calculated through SFM and BA. Finally, the graph-based method through sliding window is utilized to optimize the all the pose data. For the back-end of VINS-Fusion, the graph-based optimization part is analyzed to explore the influence on loop closure poses and number of optimized adjacent poses. The back-end framework of VINS-Fusion is shown in Fig. 4.

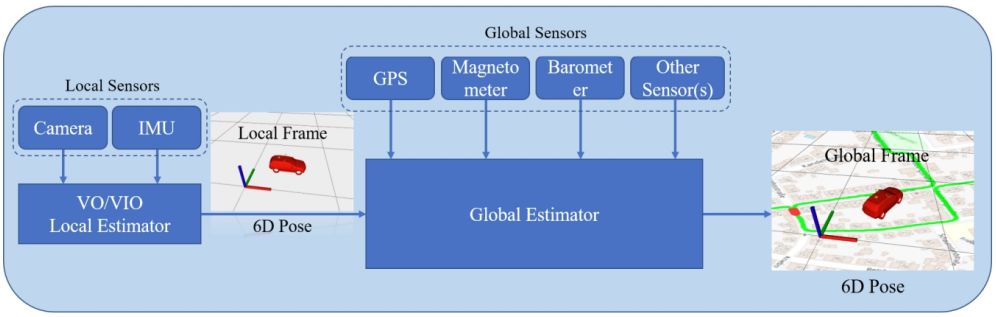

Fig. 4. VINS-Fusion Framework [5]

III. Methodology

A. ORB-SLAM

i. Front-End

To improve the efficiency and quality of detecting feature points, Bilateral Filter and Sobel operator were implemented to preprocess the image frames before inputed into ORB extractor. The Bilateral Filter is designed to smooth images while preserving edges by combining spatial proximity with intensity similarity [8], [9]. The mathematical formula for the Bilateral Filter could be express as Eqn. 8. Where I is the intensity of pixel, G_{σ_s} is the spatial Gaussian kernel, which decreases with the Euclidean distance between the two pixel locations p and q (hence $\|p-q\|$), G_{σ_r} is the range Gaussian kernel, which decreases with the difference in pixel intensity values $|I_p - I_q|$, and W_p is the normalization term calculated as the sum of the products of the two Gaussian kernels for all the pixels in the neighborhood S, which ensures that the filter preserves the energy level of the image.

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p-q\|) G_{\sigma_r}(|I_p - I_q|) I_q \quad (8)$$

Additionally, the Sobel operator estimates gradients that measure the rate of change in intensity at each pixel position in an image, indicating edge orientation and strength. According to [10], the Sobel operator applies two separate kernels to the image to compute the gradients in the horizontal (x) and vertical (y) directions. Assuming A is the original image, and I_h and I_v are intensity changes at horizontal and vertical respectively. G_x and G_y are two images which at each point contain the vertical and horizontal derivative approximations, they are computed as:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I_h \quad (9)$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I_v \quad (10)$$

After that, the gradient magnitudes can be combined to give the total magnitude of the gradient at each point:

$$G = \sqrt{G_x^2 + G_y^2} \quad (11)$$

The use of the Bilateral Filter and the Sobel operator in image preprocessing is strategic for feature detection tasks in ORB-SLAM3. The Bilateral Filter smooths the image while preserving edges, reducing the effect of noise and minor color variations. The Sobel operator then uses this cleaner image to detect and outline significant edges, crucial for feature localization and mapping tasks in subsequent stages of ORB-SLAM. These steps ensure that the input to the feature detection algorithm is optimized for both accuracy and performance.

ii. Back-End

The back-end of ORB-SLAM3 mainly uses BA to participate in constructing optimization problems and solving them. Based on the observations of geometric and algebraic relationships between camera poses and map points, the formulating error functions are generated. When incorporating the camera's intrinsic parameters and the estimated poses of the keyframes, a mapping relationships between the map point's spatial coordinates and its pixel coordinates can be established. Ideally, reprojecting the map point back on keyframes should generate the pixels corresponding to the feal feature points. Nevertheless, reprojection errors normally appear in real processing steps. To solve this, the original code uses g2o, but adding GTSAM to further finding the optimal map point coordinates and keyframes poses that minimize the sum of all these loss functions is one of the method to enhance the performance of ORB-SLAM3.

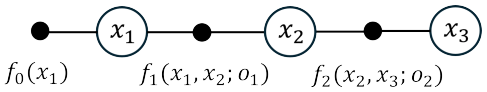

Fig. 5. Factor graph for robot localization

GTSAM is an open-source C++ library designed to optimize nonlinear error functions. The core of GTSAM is the factor graph, a category consisted of variables and factors, shown in Fig. 5. The variables x_0, x_1, and x_2 represents the real-time robots poses, while the factors, such as unary factor f0 indicating the prior acknowledgement of pose x_1, binary factors f_1 and f_2, helps to relate nearby poses. With utilization of smoothing and mapping techniques to process temporal and spatial measurements, GTSAM succeeds in evolving the state variables. In addition, GTSAM further iteratively refines the estimates of state variables to minimize the sum of squared errors through Levenberg-Marquardt algorithm. As a method which is different from the original g2o libraries, which mainly set keyframes as vertices and map points as edges to iteratively minimize the residual errors across the graph, using GTSAM to continuously update the information which has already been process via g2o can take advantages of both methods so that the ORB-SLAM3 will have a better performance than when only using g2o.

B. VINS-Fusion

i. Front-End

As introduced before, VINS-Fusion utilizes LK optical flow to track features among frames. Before that, we need to know which features to track. Thus, a Shi-Tomasi corner detection method is involved. This method calculate the score of a pixel based on its rate of change in intensity to determine if its a corner or so. The score (in Eqn. 12) comes from the smaller eigenvalue of the gradient covariance matrix (in Eqn. 13) of the pixel. The intensity of a pixel, i.e., I_x and I_y are calculated by the Sobel operator.

$$R = \min(\lambda_1, \lambda_2) \tag{12}$$

$$M = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \tag{13}$$

Although the combination of both would theoretically enhance the result of the feature detection process, considering the original structure and functions used in VINS-Fusion, it might also cause some level of drawback at the sametime. We will discuss this in detail along with the results in the discussion.

Similarly to what we did to ORB-SLAM, we added a bilateral filter followed by a Sobel operator to feed more distinct corner features to the feature tracker in VINS-Fusion.

ii. Back-End

The back-end of VINS-Fusion implemented graph-based method to optimize the pose through ceres. Graph-based method is a powerful approach to solve the SLAM problem by representing the relationships between robot poses and landmarks as a graph. In this method, the nodes of the graph represent the poses of the robot at different times or the positions of landmarks in the environment, while the edges represent the spatial constraints between these nodes, usually derived from sensor measurements like Lidar, IMU and cameras. After constructing the graph basing on the processed data, the poses and corresponding relationship is determined, preparing for pose optimization process [11].

Ceres Solver is also a robust optimization library, which can be used for graph optimization process [12]. It excels in the optimization of nonlinear least squares problems, incorporating loss functions that are crucial for accurate map construction and trajectory estimation in SLAM. In VINS-Fusion, ceres is used to model and optimize the graph. The global pose graph structure is shown in Fig. 6.

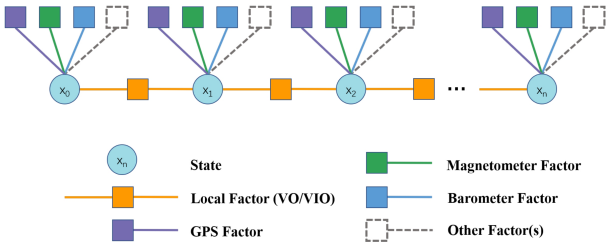

Fig. 6. VINS-Fusion global pose graph [5]

The edges in the graph is collected from adjacent poses and loop poses. In this project, the graph-based optimization performance is analyzed with more adjacent edges adding on the pose graph.

IV. Results

To evaluate the performance of both models after applied corresponding optimizations, we incorporate EuRoC dataset. Based on the level of difficulty provided in the official site, MH-01-easy and MH-05-difficult are two main datasets incorporated in experiments to test models' performance in ORB-SLAM3 and Vins-Fusion respectively with various modifications. The performance metrics under consideration encompass both positional and rotational accuracies. Positional accuracy is quantified by the maximum error (max) and the root mean square error (rms) in meters, which measure the largest

single error and the standard deviation of the error distribution, respectively. Rotational accuracy is likewise measured, represented in degrees, offering insights into the orientation precision of the frameworks. Here is the link of video presentation: https://www.youtube.com/playlist?list=PLviw48dOCAFc3h8vIsqojVLFG4Ydin4yF.

A. ORB-SLAM

Just as what has been introduced in Methodology part, both the front-end and back-end part of ORB-SLAM3 has been added to further minimizing the errors generated in SLAM process.

The Fig. 7 shown below consists of path generated in four situations: the provided ground truth, the path generated by original ORB-SLAM3; the path incorporating the GTSAM, and the path takes advantages in both GTSAM and new ORB extractors.

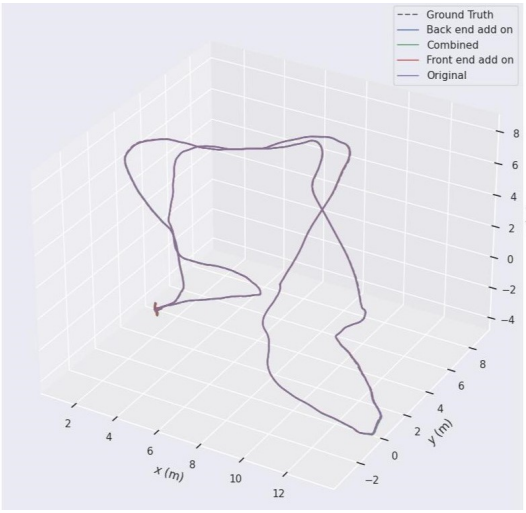

Fig. 7. ORB-SLAM3 assembling path for MH-05-difficult under stereo cameras with IMU support

Even though from the general figure, it is not clear to see the change of path under different situations when setting difficult dataset as target, there is a significant deviations when using the difficult dataset, as shown in both table 1 and table 2. Setting MH-05-difficult as target, the only Front-end add on takes -6.03% in max position errors while the only Back-end add on takes -31.6% in max position errors, representing a significant improvements in ORB-SLAM3's performance. However, when incorporating IMU support to do SLAM, the corresponding changes in error has increased to -0.019% and -0.059%, while the IMU support itself provides -24.1% in position errors, reflecting the fact that both of the optimization methods are less useful with such a strong IMU support. This result has also been indicated by the fact of reducing max rotation error from 90.076 degrees to 1.310 degrees. Furthermore, as the data restored in table shows, the performance of model with taking both new extractors and GTSAM simultaneously does not perform as good as when taking them separately, leading to an assumption that there are some conflicts taking place between both methods so that reducing the performance of ORB-SLAM3.

B. VINS-Fusion

In VINS-Fusion, MH-01-easy and MH-05-difficult datasets are also used to analyze the pose graph optimization performance. The graph is similar with ORB-SLAM3 above, containing ground truth, original, front end add on, back end add on and combined path. The MH-05 path comparison is shown in Fig. 8.

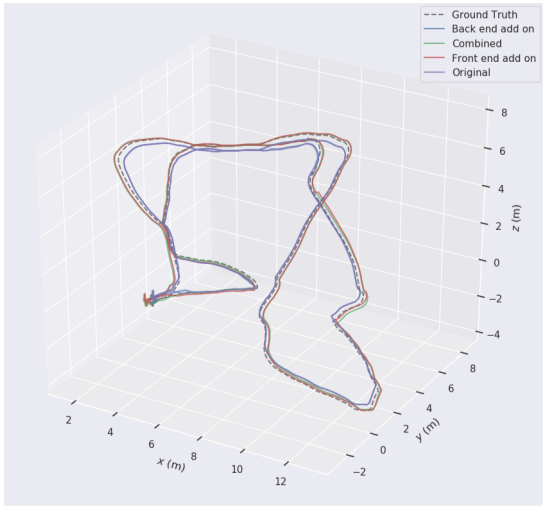

Fig. 8. VINS-Fusion assembling path for MH-05-difficult under stereo cameras with IMU support

The front end and back end add on get better optimization performance in MH-01 dataset, but have not contribute a lot in MH-05 dataset. For back end add on, it also decreases the position rms with MH-01 dataset. With respect to rotation, front end and back end add on both improve the estimation accuracy in MH-05 dataset, but have not bring better performance in MH-01 dataset.

When the stereo camera is only used with MH-01 dataset in front add on situation, the system fail to locate its pose with a large max position error over 7m and max degree error over 100 degree. The problem potential occurs at the start of the experiment. The matched number of feature points is pretty low, and integrating with a suddenly drop off of the camera, VINS-Fusion estimate its pose far away from the truth.

TABLE I
COMPARISON OF SLAM RESULTS (STEREO CAMERA)

Frameworks	Modify	MH-01-Easy				MH-05-Difficult			
		Position (m)		Rotation (deg)		Position (m)		Rotation (deg)	
		max	rms	max	rms	max	rms	max	rms
ORB-SLAM3	Original	0.115	0.040	90.138	88.836	0.199	0.047	90.076	89.061
	Front-end add on	0.105	0.036	90.440	89.021	0.187	0.055	90.092	88.963
	Back-end add on	0.093	0.033	90.284	88.966	0.136	0.049	90.545	89.044
	Combined	0.108	0.036	90.457	89.053	0.168	0.052	89.640	89.025
VINS-Fusion	Original	0.725	0.222	9.093	6.193	0.654	0.243	7.234	3.801
	Front-end add on	Fail	Fail	Fail	Fail	1.039	0.391	2.450	1.537
	Back-end add on	0.870	0.326	11.930	9.847	0.603	0.281	4.192	2.507
	Combined	Fail	Fail	Fail	Fail	0.916	0.428	2.453	1.636

TABLE II
COMPARISON OF RESULTS (STEREO CAMERA + IMU)

Frameworks	Modify	MH-01-Easy				MH-05-Difficult			
		Position (m)		Rotation (deg)		Position (m)		Rotation (deg)	
		max	rms	max	rms	max	rms	max	rms
ORB-SLAM3	Original	0.041	0.014	2.783	1.186	0.151	0.051	1.310	0.623
	Front-end add on	0.068	0.023	2.701	1.216	0.148	0.045	1.324	0.602
	Back-end add on	0.044	0.019	2.788	1.174	0.142	0.047	1.298	0.610
	Combined	0.038	0.016	2.724	1.231	0.147	0.048	1.404	0.619
VINS-Fusion	Original	0.327	0.160	5.676	1.951	0.295	0.161	4.347	1.781
	Front-end add on	0.253	0.158	8.399	2.177	0.259	0.167	7.179	1.344
	Back-end add on	0.353	0.109	8.637	2.412	0.306	0.165	4.240	1.769
	Combined	0.336	0.132	8.912	2.636	0.287	0.170	3.063	1.144

V. DISCUSSION & CONCLUSION

A. ORB-SLAM

For ORB-SLAM, modifications were implemented on both front and back ends while they both focused on visual part of the SLAM system. This consequently yielded the fact that if IMU plays a significant role in the fusion of sensors data, the improvement might not be that obvious according to the results. But when IMU's disabled, reasonable improvements in both rotation and translation were boserved. Also, the combination of front and back ends somehow reduced the amount of improvement. In the future, we might want to dive deep into how ORB-SLAM fuses its sensors data to figure out why our modification seems to be contradictory to each other.

B. VINS-Fusion

Modifications on both the front and back ends of VINS-Fusion do not necessarily generate a more robust and better outcome. Bilateral filters and Sobel operators might be good at enhancing the visibility and distinctiveness of edges and corners in complex environments, but it might also increase the weight of the noise and overshadow edges over corners, which are heavily used in Shi-Tomasi method. These withdraws might result in failure of tracking features across two frames and thus result in failure of pose estimations. But IMU won't be affect by these modification and thus can consistently generate reliable pose calculations along the trajectory.

C. Future Work

Last but not the least, there are considerable number of parameters in the bilateral filter and Sobel implementation process for both ORB-SLAM and VINS-Fusion. Unfortunately we did not have the time to experimentally come up with the "optimal" ones. To solve this problem, we could use learning based feature extraction [13] method to obtain parameters with better performance.

REFERENCES

[1] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[2] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443.

[3] P. L. Rosin, "Measuring corner properties," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 291–307, 1999.

[4] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, 2006, pp. 850–855.

[5] Q. Tong, S. Cao, J. Pan, and S. Shen, "A general optimization-based framework for global pose estimation with multiple sensors," *arXiv.org*, 2019.

[6] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, 1981. [Online]. Available: https://api.semanticscholar.org/CorpusID:2121536

[7] V. Tarasenko and D.-W. Park, "Detection and tracking over image pyramids using lucas and kanade algorithm," 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:27505652

[8] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, 1998, pp. 839–846.

[9] R. G. Gavaskar and K. N. Chaudhury, "Fast adaptive bilateral filtering," *CoRR*, vol. abs/1811.02308, 2018. [Online]. Available: http://arxiv.org/abs/1811.02308

[10] N. Kanopoulos, N. Vasanthavada, and R. Baker, "Design of an image edge detection filter using the sobel operator," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 2, pp. 358–367, 1988.

[11] Q. Tong, S. Cao, J. Pan, and S. Shen, "A general optimization-based framework for global pose estimation with multiple sensors," *arXiv.org*, 2019.

[12] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 10 2023. [Online]. Available: https://github.com/ceres-solver/ceres-solver

[13] D. Lunga, S. Prasad, M. M. Crawford, and O. Ersoy, "Manifold-learning-based feature extraction for classification of hyperspectral data: A review of advances in manifold learning," *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 55–66, 2014.

[14] "Factor graphs and gtsam," *GTSAM*, 2019. [Online]. Available: https://gtsam.org/tutorials/intro.html