Project Report: SLAM
Vision-Based Navigation

Codebase: https://github.com/AlboAlby00/Slam
Supervisor: Sergei Solonets

Alberto Confente
Technical University of Munich, Germany
Email: albertoconfente2000@gmail.com

Priyadharshinee Selvan
Technical University of Munich, Germany
Email: spriyadharshinee@gmail.com

Abstract

This project extends an Indirect Visual Odometry codebase to incorporate Simultaneous Localization and Mapping (SLAM) by implementing additional data structures necessary for effective map management. These data structures enable the representation of spatial relationships between keyframes by connecting those that share common visual features, allowing for efficient optimization through Bundle Adjustment on just a targeted subset of all the landmarks and keyframes. Our approach was able to significantly increase the accuracy of the localization with only a minor increase in the computational cost of the algorithm.

I. Introduction

A. Visual Odometry Limitations

Visual Odometry (VO) estimates a robot's or camera's motion by analyzing sequential images, retaining only a queue of the most recent keyframes and discarding older ones.
This approach is computationally efficient because it allows Bundle Adjustment to be performed only on a limited number of keyframes and landmarks. However, it has some inherent limitations. Over time, small errors in motion estimation can accumulate, resulting in cumulative drift. This drift causes inaccuracies in positioning, especially in long trajectories. Additionally, as older keyframes are discarded, all past information about a location is lost after a time. This results in treating a location visited in the past in the exact same way as locations that are completely new. To be able to exploit the information about past locations, a trivial approach would be not to discard any keyframe, but this approach quickly becomes computationally infeasible.

B. Our Implementation Overview

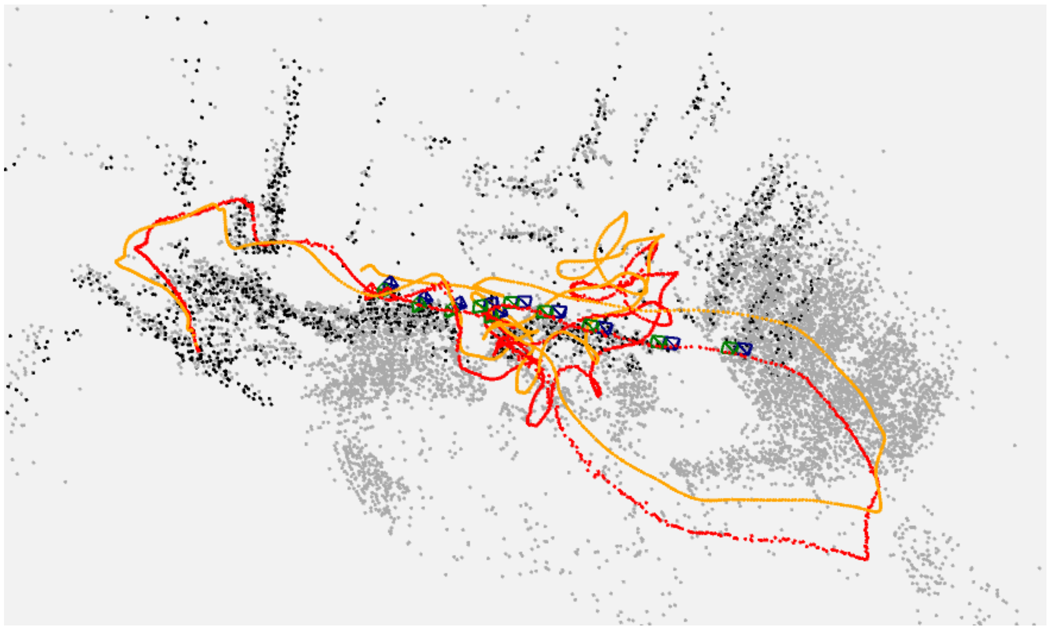
Our approach succeeds in finding a balance between these two extreme approaches, being able to exploit past data at an acceptable computational price.
Once a new keyframe is added, new landmarks are added to the map. Cameras that do not have many landmarks in common with the new keyframe will not have any substantial improvement from bundle adjustment. We exploited this fact. Bundle Adjustment is run only on the subset of the total keyframes that have a minimum number of landmarks in common with the new keyframe and on the landmarks visible from these keyframes. To implement this approach, some additional data structures had to be implemented. Section II will have an extensive description of it.

Fig. 1 illustrates trajectory estimations using both Visual Odometry and our SLAM implementation on the Euroc V1 Easy dataset [1]. Unlike VO, our approach retains past keyframes (depicted as black cameras), but bundle adjustment is applied only on a small localized subset of cameras (shown as green and blue cameras). Fig. 2 shows the plots of error difference between the ground truth and estimated trajectories for these two methods.

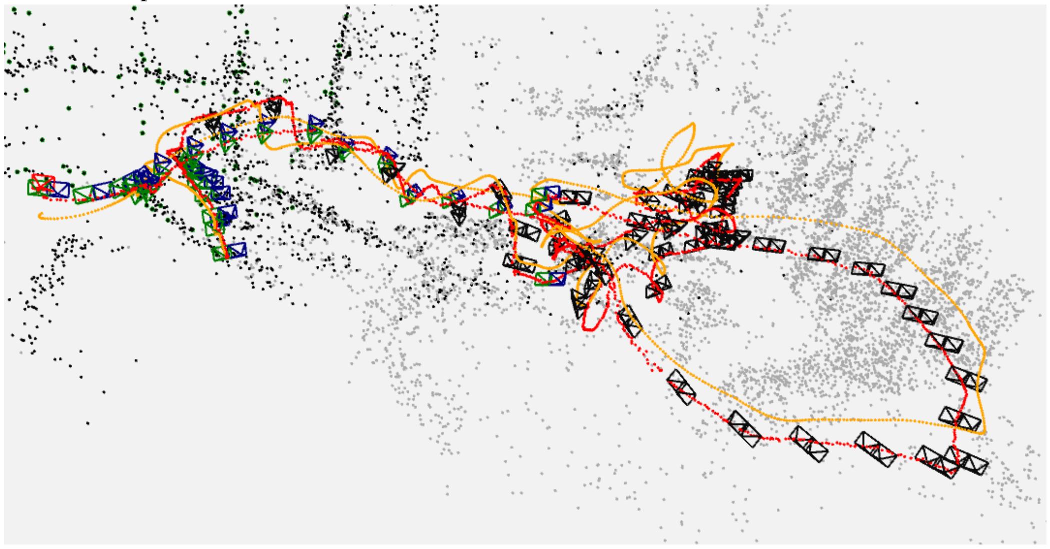
II. Data Structures

A. Global Map

The set of all the landmarks detected.



(a) Using the Visual Odometry codebase. Past keyframes are discarded and cannot be used if we revisit a past location.

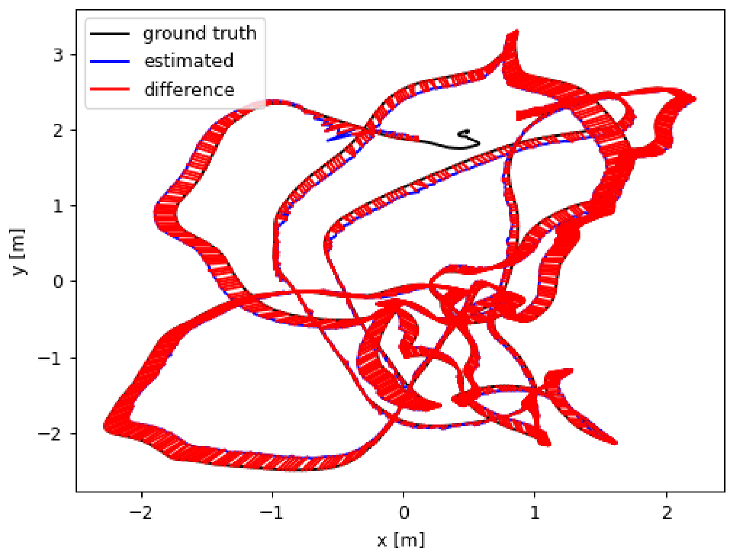


(b) Using our SLAM codebase. No keyframe is discarded. Bundle adjustment is run on the blue and green cameras and black landmarks and is not run on black cameras and gray landmarks.

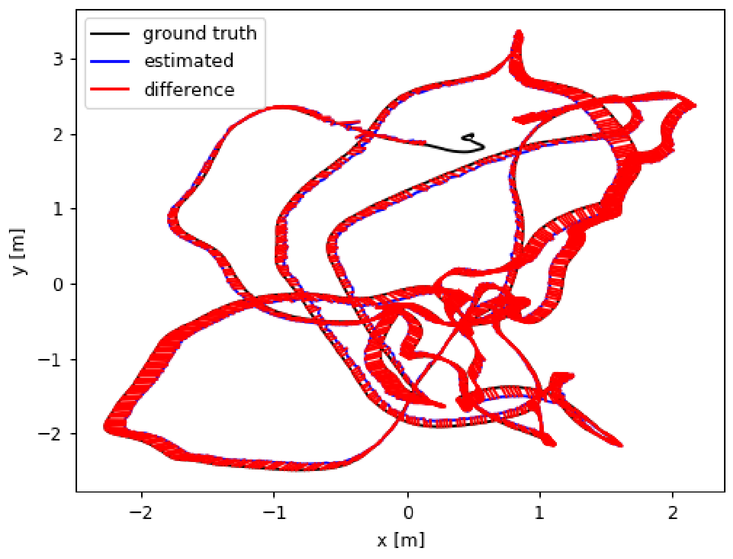
Fig. 1: Comparison of trajectory estimation on Euroc V1 Easy Dataset [1] using different approaches. Ground truth trajectory is visualized in yellow, and the estimated trajectory is in red.

B. *Covisibility Graph*

The covisibility graph is a data structure used to store information about the different keyframes that share common landmarks. In this graph, keyframes are represented as nodes, and an edge is created between two nodes if the corresponding keyframes have a minimum number of landmarks in common.
Each time a new keyframe is added, a corresponding node is added to the covisibility graph. The global map is then projected on this new keyframe. By iterating through all existing keyframes, the necessary edges are added to update the covisibility graph accordingly. Fig. 3 shows an illustration of a covisibility graph, as depicted in ORB-SLAM2 [2]



(a) Using Visual Odometry.



(b) Using our SLAM implementation with local optimization.

Fig. 2: Error between the ground truth and estimated trajectory on the Euroc V1 Easy dataset [1]

C. Local Map

Each keyframe is associated with a local map. The local map for a keyframe includes all the landmarks visible in that keyframe, as well as the landmarks visible in all keyframes connected to it within the covisibility graph. The local map of the most recent keyframe is used to localize all frames that are not keyframes. This approach is more efficient than projecting the entire global map, as it reduces computation time while maintaining localization accuracy.

Unlike in Visual Odometry, where past keyframes are discarded, using the local map allows for the utilization of landmarks that were created in earlier frames. This capability helps mitigate the drifting problem, particularly when

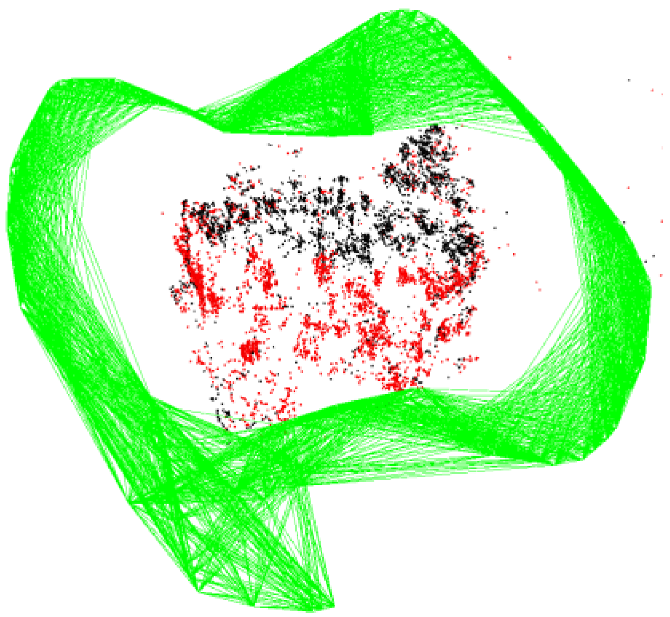


Fig. 3: Covisibility graph visualization taken from the ORB-SLAM2 paper [2]

the camera revisits previously explored locations.

III. Local Bundle Adjustment

As mentioned previously, running the Bundle Adjustment (BA) on all keyframes is computationally infeasible. Instead, each time a new keyframe is added, we run only a local bundle adjustment. Local Bundle Adjustment uses only the keyframes that have a minimum number of landmarks in common (and so they are connected in the covisibility graph) and it uses only the landmarks in the local map. It makes sense to do this since bundle adjustment will have a big impact only on the cameras that are facing the newly created landmarks, and will have a small or no influence on all the other keyframes.

Local Bundle Adjustment is handled as a separate task from the tracking process and is therefore executed on a dedicated thread to ensure efficient computation and minimize interference with the real-time tracking.

IV. KeyFrame Management

The ORB-SLAM paper [3] attributes much of its robustness to the ”survival of the fittest” strategy employed in keyframe management. This involves inserting keyframes as quickly as possible and later removing unnecessary ones to minimize computational overhead. We've implemented a similar strategy to evaluate its impact on the performance of our SLAM system.

A. KeyFrame Insertion Strategies

We evaluated two additional keyframe insertion criteria alongside the existing strategy used in the Odometry codebase. When any of these three criteria are applicable, a new keyframe is added to the local map:

1) **Minimum number of Feature correspondences**
   When the number of inlier feature matches of the current frame to the landmarks in the local map is below a (hyperparameter) threshold, insert a new keyframe. This is adapted from the existing strategy used in Odometry.

2) **Maximum number of non-Keyframes**
   This imposes a limit on the maximum number of non-keyframes that can exist between two keyframes. Insert a new keyframe if the last keyframe was inserted more than x frames ago.
3) **Mean Square Optical Flow**
   If the field of view changes significantly (based on a threshold) from the previous frame to the current image, insert a new keyframe. This is measured using the mean square optical flow, as proposed in the DSO method [4]. This is given by:

$$f := \left(\frac{1}{n} \sum_{i=1}^{n} \|\mathbf{p}_i - \mathbf{p}_i'\|^2 \right)^{\frac{1}{2}} \tag{1}$$

   where n is the number of feature points tracked by both images, p_i is the position of the i^{th} feature in the previous frame, and p_i' is the corresponding position of the i^{th} feature in the current frame.

B. *Culling Redundant Keyframes*

Culling redundant keyframes is a necessary step in ensuring the local map does not grow unbounded, and managing the complexity of the local bundle adjustment. This involves discarding keyframes that do not contribute significantly to the map's accuracy. If the size of the local map is larger than m (a hyperparameter) keyframes, similar to ORB-SLAM [3], we discard all such keyframes in the local map where 90% or more of their keypoints were observed in at least k (a hyperparameter) other keyframes within the local map. These discarded keyframes are also removed from each landmark's observation and from all covisibility graphs. If all visible observations of a landmark were removed during this process, such landmarks are also removed from the overall global set of landmarks.

V. Other Features Implemented

A. *Uniform Feature Extraction*

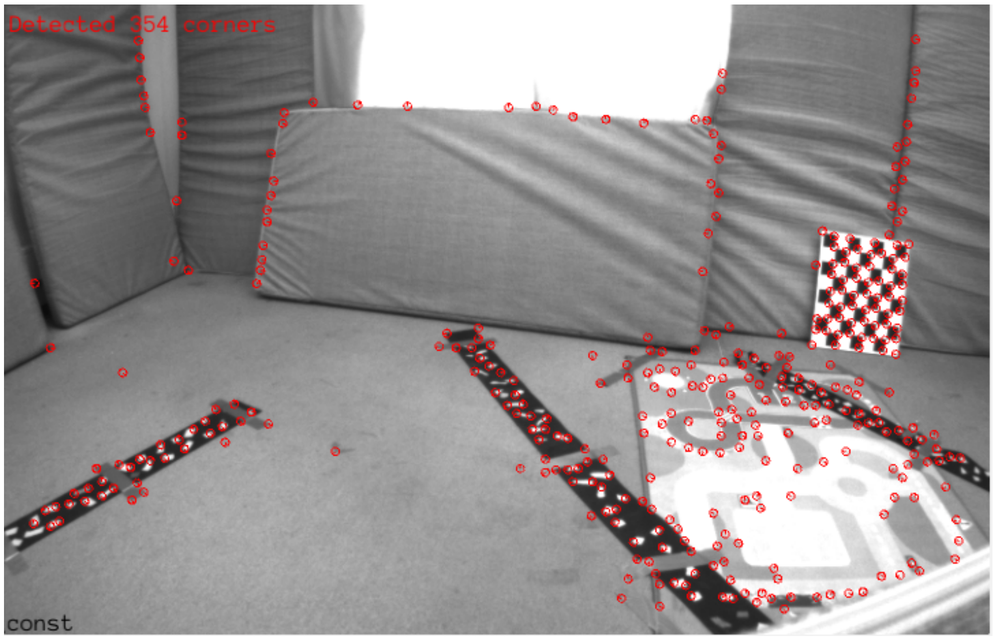
The default feature extraction method extracts most features in few specific parts of the image, ignoring others. In order to obtain a more uniform distribution of feature locations, we divide the image in a grid like it is done in the ORB-SLAM paper [3], and every cell of the grid is treated independently in the feature extraction process. Fig. 4a shows the features extracted from an image using default (non-uniform) feature matching, and Fig. 4b shows the features extracted from each grid of the same image using uniform feature matching.
When evaluating this feature, we found that the mean absolute translational RMSE value was similar to that obtained without uniform feature extraction but with a noticeably higher standard deviation.

B. *Dynamic Matching Radius*

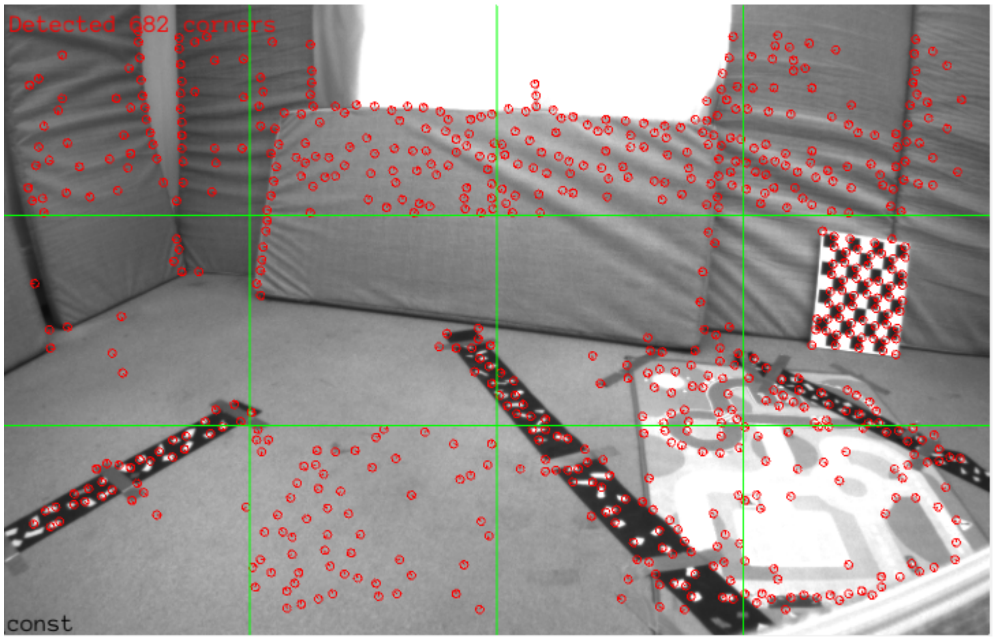
In the Visual Odometry codebase, in order to localize every new frame, the new frame pose is initialized to the old frame pose, features are extracted and landmarks are projected to the new frame. The frame pose is obtained using the matches between extracted features and the projected landmarks.
To match a projected landmark to the detected features, only the features in a circular area of constant radius around the projected landmark are considered as candidates for matching. This approach is obviously less computationally expensive compared to brute-force matching with all extracted features, but it can lead to failure in case of a too-aggressive camera movement since the extracted feature relative to the projected landmark would be outside of the circular area.
We substituted the constant matching radius with a dynamic radius calculated using the following formula.

$$\text{radius} = \text{mean\_optical\_flow} \times \text{constant} + \text{minimum\_radius}$$

This causes the radius to increase when the movement is aggressive and decrease when the movement is slow.
Despite the localization estimate still being not entirely stable, the tracking failures were substantially decreased on the Euroc dataset "Machine Hall 03 Medium" [1], where the images are recorded using a drone performing aggressive movements. Fig. 5a shows tracking failures on the "Machine Hall 03 Medium" dataset when using a constant matching radius, and Fig. 5b shows the same tracking while using a dynamic matching radius, with no tracking failures or discontinuities.



(a) Non-uniform feature extractions

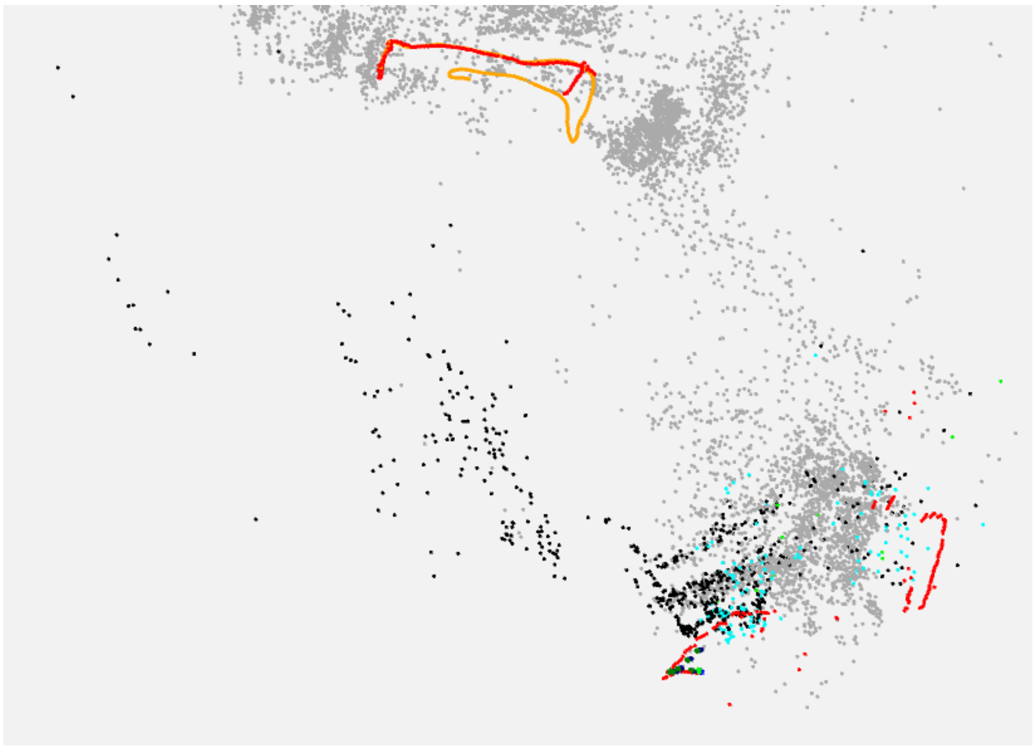


(b) Uniform feature extractions

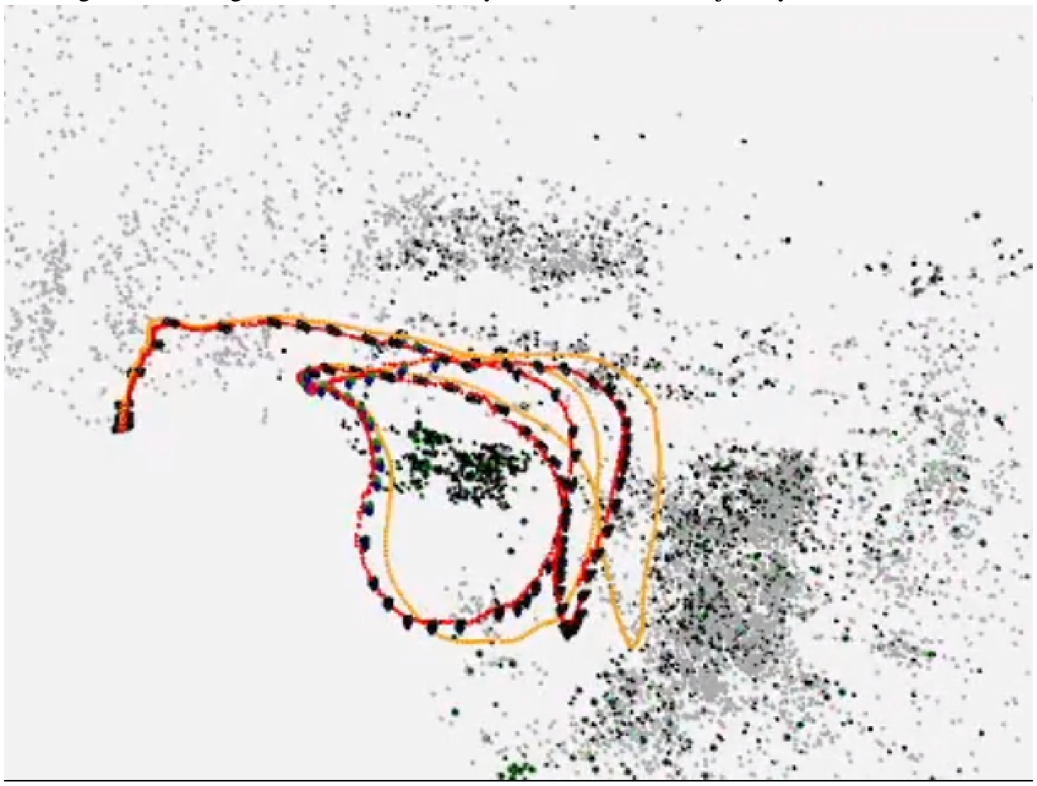
Fig. 4: Comparison of feature extraction methods

VI. Observations and Results

Once all frames have been processed by our SLAM system and the bundle adjustments are complete, we output the final optimized camera poses for each processed frame to a `.csv` file. We then evaluate the Absolute Trajectory Error (ATE) by comparing the ground truth trajectory with our estimated trajectory. The script used for computing the ATE has been adapted from the evaluation script provided in the SVO codebase [5].



(a) Using a constant matching radius. It is visible that shortly after the start of the video stream, tracking fails, causing a sudden discontinuity in the estimated trajectory.



(b) Using dynamic matching radius. In this case, at the start of the video stream, no discontinuities happened.

Fig. 5: SLAM trajectory tracking on the "Machine Hall 03 Medium" dataset [1] using different feature matching radius techniques.

A. Keyframe Management

After conducting multiple runs with each keyframe insertion strategy, we observed that neither of the two newly implemented strategies — maximum number of non-keyframes and mean square optical flow — produced statistically significant improvements in the performance of the SLAM system. The previously implemented strategy based on the minimum number of feature correspondences performed effectively across different scenarios. Since the dataset(s) that we mainly tested on do not contain any sharp turns or strong rotational movements, we did not benefit much from the mean square optical flow keyframe insertion criteria.

Culling redundant keyframes is designed to limit the complexity of local bundle adjustment, with the expectation of a slight trade-off in accuracy. To evaluate the impact of this approach on SLAM system performance, we ran Local Bundle Adjustment on the same (main) thread instead of running it in parallel on a different thread. Table I presents the results from five such runs on the Euroc `V1_01_easy` sequence, comparing the run time (in seconds) and ATE-RMSE (in meters), with and without redundant keyframe culling. [1]

Run	**Without KF Culling**		**With redundant KF culling**	
	time taken (in s)	ATE-RMSE (in m)	time taken (in s)	ATE-RMSE (in m)
1	480.016	0.099780	289.923	0.092906
2	499.199	0.097786	307.448	0.088909
3	531.378	0.106960	300.975	0.092961
4	372.000	0.097503	305.773	0.091895
5	364.291	0.108865	285.211	0.091664

TABLE I: Results of 5 runs on Euroc V1_01_easy dataset with and without Redundant Keyframe Culling.

Statistical analysis of the above results, as shown in Table II, suggests that there is strong evidence that culling redundant keyframes is both time efficient and also leads to a more accurate localization.

Metric	t-statistic	p-value
Time	4.584	0.0102
ATE-RMSE	4.777	0.0088

TABLE II: Statistical Analysis of Runtime and ATE-RMSE with and without Keyframe Culling

It is also crucial to acknowledge that these results are highly sensitive to the various hyperparameters used in keyframe insertion and culling. For the experiments described above, the hyperparameter values used were $x = 20, m = 20, k = 3$. That is, if the last keyframe was added more than 20 frames ago, a new keyframe is added and if the size of the local map is ≥ 20 keyframes, we cull redundant keyframes where $\geq 90\%$ of keypoints were observed in at least 3 other keyframes in the local map.
A significant challenge encountered during these experiments was that overly aggressive addition or culling of keyframes sometimes led to tracking failures. These failures were due to the disproportionate influence of certain keyframes or insufficient data. To maintain the integrity of the reported results, we excluded such outlier data from the final outcomes to minimize noise and ensure more accurate performance evaluation.

B. Local Bundle Adjustment

As said previously, the Local Bundle Adjustment improved the localization quality. Here we compared the Odometry codebase and our SLAM implementation that resulted in the best performance. We didn't cull any

[1]*Note: In the Euroc V1_01_easy dataset, the drone remains stationary for the first approximately 100 frames. To prevent multiple local Bundle Adjustment runs and repetitive keyframe insertion and culling during this period, we disabled both local bundle adjustment and keyframe culling for the first approximately 100 frames in this specific experiment.

keyframe, we used the default keyframe selection strategy and we didn't use the uniform feature extraction.
We evaluated the algorithms on the ”Euroc MAV Vicon Room 2” dataset [1] using the Absolute Translational Root Mean Squared Error as a metric. Both codebases are non-deterministic due to multithreading and due to RANSAC algorithm, so multiple runs are required to compare them. Table III shows the results of 10 such runs, and Fig. 6 shows the confidence levels and margin of error from these runs.

Run	Odometry ATE-RMSE (m)	SLAM ATE-RMSE (m)
1	0.127268	0.105565
2	0.107698	0.106949
3	0.139505	0.109412
4	0.132889	0.101339
5	0.101372	0.104475
6	0.134385	0.107137
7	0.113120	0.101770
8	0.126611	0.102170
9	0.125374	0.104074
10	0.115301	0.099738

TABLE III: Results of 10 runs on the Eurco MAV Vicon Room 2 dataset using Odometry and our SLAM approach.

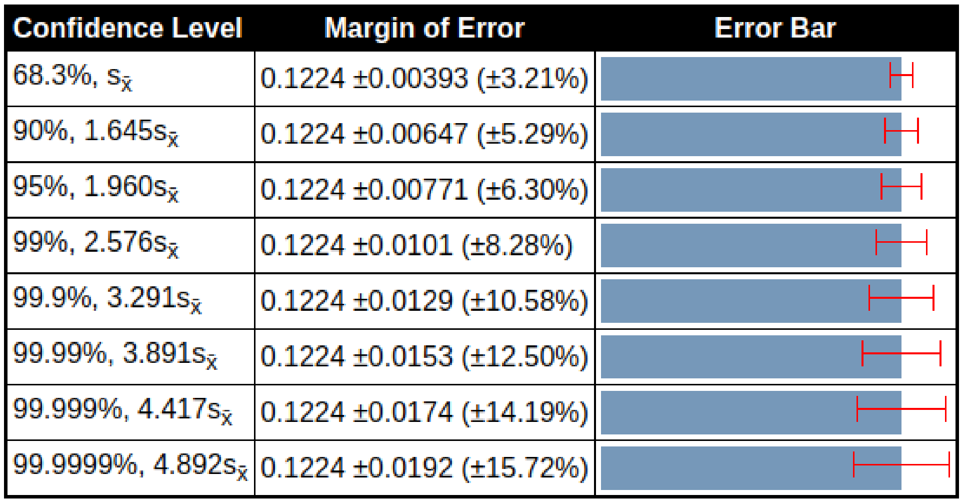
VII. Conclusion

In this project, we successfully extended our Indirect Visual Odometry codebase to incorporate a few significant SLAM capabilities by implementing critical data structures for map management. While Visual Odometry discards older keyframes and recovers the path incrementally to focus on the local consistency of the trajectory, information that can be gained from previously processed keyframes and visited landmarks, which can contribute to the global map consistency, is permanently lost. By building a covisibility graph, we enabled the system to focus on the local trajectory consistency, similar to Visual Odometry, and perform bundle adjustment on just the local map, enhancing the efficiency, while also retaining the set of older keyframes and landmarks for future referencing.

Throughout the project, we implemented and tested several key components:

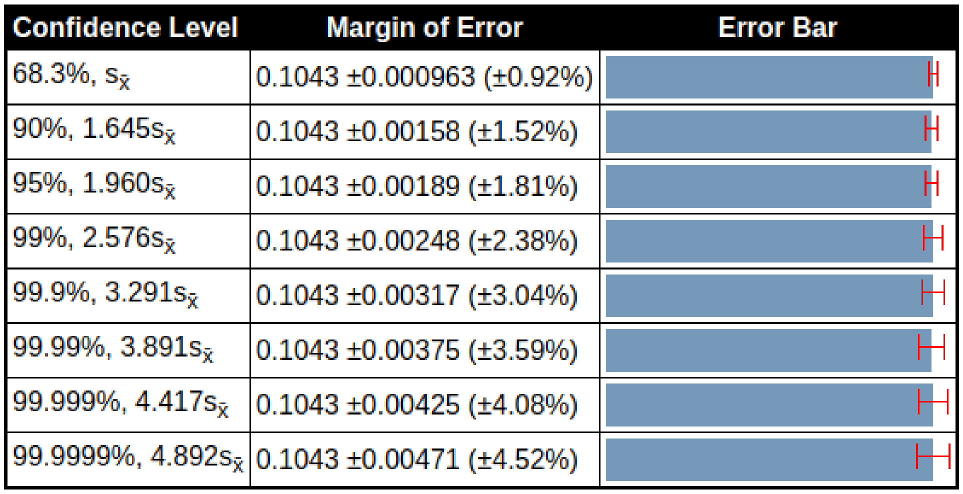
1) **Local Map and Local Bundle Adjustment:** These were crucial for improving the accuracy of the SLAM system. By focusing on a local map, we were able to limit the scope of computationally expensive operations, striking a balance between precision and performance.
2) **Keyframe Insertion Strategies:** We explored different strategies for inserting keyframes, which is vital for maintaining an up-to-date and accurate map. Our results indicated that while different strategies have varying impacts, the initial approach used in VO proved robust across various scenarios.
3) **Culling Redundant Keyframes:** We implemented a strategy for removing redundant keyframes, which helped to manage the complexity of local bundle adjustment. This culling process provided both a slight speed advantage and a positive impact on overall accuracy, in most scenarios.
4) **Feature Matching:** We introduced a dynamic matching radius that adjusts based on the mean optical flow, which helped mitigate tracking failures due to aggressive camera movements. We also implemented and tested a uniform feature extraction technique by dividing the image into equal grids to ensure a more even distribution of features. However, this did not substantially change the ATE-RMSE results.

The culmination of these efforts resulted in a SLAM system that significantly improved the accuracy of localization with only a minor increase in computational costs.



Confidence Level	Margin of Error	Error Bar
68.3%, $s_{\bar{x}}$	0.1224 ±0.00393 (±3.21%)	
90%, 1.645$s_{\bar{x}}$	0.1224 ±0.00647 (±5.29%)	
95%, 1.960$s_{\bar{x}}$	0.1224 ±0.00771 (±6.30%)	
99%, 2.576$s_{\bar{x}}$	0.1224 ±0.0101 (±8.28%)	
99.9%, 3.291$s_{\bar{x}}$	0.1224 ±0.0129 (±10.58%)	
99.99%, 3.891$s_{\bar{x}}$	0.1224 ±0.0153 (±12.50%)	
99.999%, 4.417$s_{\bar{x}}$	0.1224 ±0.0174 (±14.19%)	
99.9999%, 4.892$s_{\bar{x}}$	0.1224 ±0.0192 (±15.72%)	

(a) Margin of error table calculated using the Odometry RMSE



Confidence Level	Margin of Error	Error Bar
68.3%, $s_{\bar{x}}$	0.1043 ±0.000963 (±0.92%)	
90%, 1.645$s_{\bar{x}}$	0.1043 ±0.00158 (±1.52%)	
95%, 1.960$s_{\bar{x}}$	0.1043 ±0.00189 (±1.81%)	
99%, 2.576$s_{\bar{x}}$	0.1043 ±0.00248 (±2.38%)	
99.9%, 3.291$s_{\bar{x}}$	0.1043 ±0.00317 (±3.04%)	
99.99%, 3.891$s_{\bar{x}}$	0.1043 ±0.00375 (±3.59%)	
99.999%, 4.417$s_{\bar{x}}$	0.1043 ±0.00425 (±4.08%)	
99.9999%, 4.892$s_{\bar{x}}$	0.1043 ±0.00471 (±4.52%)	

(b) Margin of error table calculated using the SLAM RMSE

Fig. 6: Comparison of margin of error tables for Odometry and SLAM

References

[1] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *Int. J. Rob. Res.*, vol. 35, no. 10, p. 1157–1163, sep 2016. [Online]. Available: https://doi.org/10.1177/0278364915620033

[2] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[4] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," 2016. [Online]. Available: https://arxiv.org/abs/1607.02565

[5] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 15–22.