Collaborative SLAM with Convolutional Neural Network-based Descriptor for Inter-Map Loop Closure Detection

Zuyuan Zhu*, Zakaria Chekakta*, Nabil Aouf*,
* City, University of London, United Kingdom
Emails: zuyuanzhu@gmail.com, zakaria.chekakta@city.ac.uk, nabil.aouf@city.ac.uk

***Abstract*—This paper introduces a novel Collaborative Simultaneous Localization and Mapping (CSLAM) framework, enhanced with a Histogram of Oriented Gradients (HOG) descriptor, to improve Inter-Map Loop Closure Detection. Our framework stands out by integrating a convolutional neural network-based loop closure detection, employing the HOG descriptor for enhanced illumination robustness, and utilizing collaborative mapping from multiple robotic agents for refined pose estimations and mapping precision. Tested in diverse real-world fields, particularly for landmine detection, the framework demonstrates superior robustness and accuracy, outperforming the existing CCM-SLAM model. Additionally, it incorporates a transformation matrix from visual SLAM for LiDAR Point Clouds correction, showcasing its efficacy in 3D mapping and localization in GNSS-denied settings. Our results indicate that incorporating the CALC descriptor within a CSLAM system significantly enhances loop closure detection and mapping precision, marking a significant step forward in autonomous cooperative navigation and mapping in challenging environments**

***Keywords*—visual SLAM, dynamic environment, pose estimation, robot vision systems**

I. Introduction

Positioning, Navigation, and Mapping (PNM) play a vital role in high-risk operations, where the level of risk is greatly increased and the margin for error is unusually small. In hazardous situations, such as those encountered during disaster response, search and rescue and defense operations, the importance of rapid, accurate and reliable PNM is self-evident. When traveling through unknown and dangerous terrain, developing precise and safe routes is key to ensuring mission safety and success. In such a high-risk environment, the ability to independently develop offline and real-time mission plans becomes critical. Visual simultaneous localization and mapping (SLAM), with its capability to construct detailed 3D maps and estimate camera ego-motion, significantly enhances PNM capabilities by providing dynamic insights into unknown environments.

Visual SLAM is a key technique that has been receiving significant interests from computer vision, robotics, and augmented reality communities [1], [2], [3]. With one or multiple video cameras, visual SLAM can be used for estimating the ego motion of the camera itself and reconstructing the 3D map of the usually unknown environments. Both the ego motion and the 3D map are critical to marker-less augmented reality, autonomous navigation for robotics. Comparing with SLAM techniques using other sensors like LiDAR [4], a visual SLAM system uses video cameras which are usually light, cheap, and contains rich visual information, and is more suitable for platforms of low cost and limited payload. However, visual SLAM also faces challenges such as sensitivity to dynamic lighting conditions, susceptibility to motion blur, and the need for high computational resources, which can sometimes limit its efficacy [5].

With state-of-the-art SLAM systems having reached substantial robustness and accuracy in the centimeter range for single-robot applications, multi-robot systems have been gaining growing popularity in numerous scenarios, ranging from search-and-rescue applications [6] to digitization of archeological sites [7]. Increasing the robustness of the system by sharing information amongst the participants, boosting the efficiency of a mission by dividing up a task or enabling tasks otherwise impossible for a single robot are only some of the advantages a team of robots easy to offer. At the same time, SLAM systems designed for multiple agents have gained significant interest. Historically, much of this research centered on utilising range sensors, such as lasers and sonars, for multi-agent SLAM [8]. These early systems leveraged relative measurements, both in terms of range and bearing, between different agents to estimate the collective state of all participating entities. With the rapid advancement and increasing robustness of monocular visual SLAM, the logical progression has been to explore its application to multi-camera setups, especially those involving a team of independently moving cameras. Implementing Collaborative SLAM enhances the PNM capabilities of individual agents by pooling spatial insights and compensating for individual limitations, ultimately leading to a richer, more detailed, and more accurate mapping and localization process.

This paper focuses on using collaborative robotics technology to effectively navigate through high-risk areas. Robots working collectively can enhance the ability to assess, adapt and act in complex situations, extending range and increasing the precision of operations. Through collaborative efforts, robots can share information and insights, enabling greater situational awareness and more effective decision-making processes. By employing Collaborative SLAM (CSLAM), this paper aims to enable robots to jointly build coherent and unified maps without Global navigation satellite system

(GNSS) while positioning themselves within these maps. This not only facilitates accurate map loop detection, but also facilitates the synchronization of diverse data sets, allowing harmonious integration of individual robot perceptions into a unified environmental understanding.

II. Methodology

A. Convolutional Autoencoder for Loop Closure

The convolutional autoencoder for loop closure (CALC) architecture is a cutting-edge approach to identifying inter-agent trajectory loops [5]. This system leverages the capabilities of the convolutional neural network and HOG feature, especially when processing images from multi-agents such as SUGV and LUGV with differing viewpoints. At its core, the loop closure detection (LCD) employs the CALC model to extract global HOG features from the input images.

Figure 1 showcases the CALC neural network architecture designed for generating compact and reliable feature descriptor for visual loop closure, leveraging the principles of autoencoders and HOG for its unique design. The network accepts grayscale images of 120x160 pixels as input, which undergoes a transposition operation to rearrange its dimensions. The architecture comprises three main convolutional layers. The first layer uses 64 filters of size 1x5x5, followed by a ReLU activation, max pooling, and local response normalization (LRN). The second convolutional layer employs 128 filters of size 64x4x4, accompanied by similar post-processing steps as the first layer. The final convolutional layer utilizes 4 filters of dimensions 128x3x3, subsequently activated by a ReLU function. The output from this layer is then flattened from a 3D tensor into a 1D tensor, making it suitable to be used as a descriptor with 1064 dimensions.

To bolster efficiency and ease of use in the LCD module, the CALC model is transposed into the Open Neural Network Exchange (ONNX) format, streamlining the process of image descriptor extraction.

1) Synchronization of Image and CALC Descriptor: The LCD module is not only responsible for detecting loop closure but also for extracting the CALC descriptor, which is used to facilitate the matching of loop closure candidates, replacing the ORB-BoW. Each image (or frame) has its local ORB features, as well as global CALC features. Since the CALC descriptor and the image are published on different topics, they need to be synchronized before proceeding to the tracking stage. The synchronization between images and the CALC descriptor, as implemented in CSLAM, ensures the synchronous association of images with their corresponding CALC descriptors, even in asynchronous delivery scenarios. This is essential for maintaining data integrity and ensuring accurate tracking outcomes.

The system continuously receives images from cameras and CALC descriptors, potentially in an asynchronous manner. Therefore, a mechanism is needed to ascertain that every image is processed with its pertinent descriptor. The absence of a synchronized association could lead to incorrect processing and compromised SLAM outcomes.

Upon the receipt of an image, the system first checks if its corresponding CALC descriptor is already available. If the descriptor is unavailable, the image is temporarily stored in a buffer, denoted as mImageBuffer, pending its descriptor's arrival. Conversely, when an ONNX descriptor is received, the system checks the mImageBuffer to ascertain if the associated image is already buffered. If the image is located, the system immediately associates the image with its descriptor.

The architecture also incorporates a reset feature, allowing the entire system to be refreshed, clearing both the image buffer and descriptor associations. This feature is essential for maintaining the system's robustness, especially in scenarios where realignment or recalibration is necessitated.

The proposed association mechanism in CSLAM assures that each image is coupled with its correct CALC descriptor. This unique pairing is fundamental for achieving precise and feature-rich tracking in the SLAM system.

B. Image Masking for Enhanced Tracking

In the area where image depicts a wide low contrast grassland or goundroad, which takes up about half the image size, with trees in the distance, and a blue sky overhead.

- ORB features not work as well in low contrast scenes like grasslands or ground roads, especially when these take up a large part of the image.
- Many of the features could be getting detected on the non-informative grassland or groundroad area, leaving fewer features for the more informative parts of the scene (houses, trees).
- The environment contains many repetitive or similar patterns (like rows of trees or identical houses), this can cause incorrect feature matches, leading to errors in pose estimation.

As we know which regions of the image are less informative , we create a mask to avoid detecting features in those regions. A mask is simply a binary image, where the regions of interest are marked as "1" (or white) and the regions we want to ignore are marked as "0" (or black). We create a mask that only considers the upper half region where the trees are. The ORB detector then only detects features within this region. By doing so, we get enough high quality ORB features to detect and describe keypoints in images.

In order to further improve the tracking performance, the parameters for the ORB extractor and the tracking are also reasonably set. Based on the image scene as described above we also need to adjust other parameters to improve the number of feature matches. First of all, given the image contains large homogeneous areas (trees and shadows), increasing the number of features could help. This is because there might be fewer detectable features in these areas.

C. Replacement of BoW with CALC Descriptor

The Bag-of-Words (BoW) model is a simplified representation used to transform raw image data to vectors (in this case, feature vectors) that capture the relevant aspects of the images for the task at hand, such as matching images or recognizing

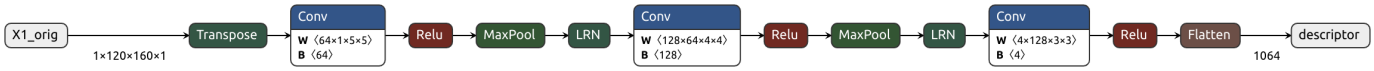

Fig. 1. The neural network architecture of the CALC model.

places. In CCM-SLAM [9], when a new image comes in, ORB features are extracted, and the BoW model is used to represent these features as a vector. This BoW vector is then compared with vectors from previous images to identify possible matches or recognise known places. The DBoW2 library is used to implement the BoW model and perform the comparison.

In our proposed methodology, we've enhanced the conventional ORB-BoW based keyframe matching technique with a preliminary check using global descriptors derived from an CALC model. This hybrid approach aims to reduce computational overhead and improve the accuracy of matching by filtering out dissimilar keyframes at an early stage.

The process initiates by extracting ONNX descriptors for a pair of keyframes. A cosine similarity measure is then used to gauge the likeness between these global descriptors. If the similarity doesn't exceed a predefined threshold, the keyframes are immediately deemed dissimilar, and the function terminates. This early exit mechanism ensures that only keyframes with a high likelihood of containing corresponding features proceed to the more computationally intensive ORB matching stage. Upon surpassing the initial similarity check, conventional ORB matching ensues. For each descriptor in the reference keyframe, we seek the closest descriptor in the target keyframe using the Hamming distance. To filter out ambiguous matches, we implement Lowe's ratio test, comparing the distance of the best match to that of the second-best. Matches satisfying this criterion are stored, and if orientation consistency is deemed crucial, a histogram-based approach is used to retain matches with the most common orientation differences.

In the CSLAM system, we transitioned from the ORB-BoW (Bag-of-Words) approach, favouring the CALC-based descriptor for its real-time performance and enhanced reliability. The places where the traditional ORB-BoW has been replaced are enumerated as follows:

- **Keyframe Matching** Inter-Map: We use the CALC descriptor for rapid matching between keyframes across different maps. This ensures a more seamless integration of separate mapped areas. Intra-Map: Similarly, for keyframes within the same map, the CALC descriptor facilitates swift and accurate matching, outperforming the traditional ORB-BoW.
- **Verification Before Sim3 Transformation** Prior to estimating the Sim3 (Similarity in 3D) transformation between the current KeyFrame and a potential loop-closing KeyFrame, we utilise the CALC descriptor for secondary verification. This step ensures that the chosen KeyFrame pairs are genuine candidates for loop closure, thus potentially reducing the number of false positives.
- **Tracking Phase** In the tracking phase of reference keyframe, we use CALC descriptor matching instead of ORB matching to find candidate for the reference keyframe. If a substantial number of matches are found, we proceed to establish a Perspective-n-Point (PnP) solver.

By integrating the CALC descriptor in these key components, we aim to leverage its superior performance in comparison to generic, off-the-shelf networks. The streamlined CALC module ensures that the descriptor extraction remains efficient and conducive to real-time operations.

D. Integrated System

Figure 2 illustrates the cohesive integration of CALC-LCD and CSLAM. In this configuration, the CALC module within CALC-LCD processes frames from both SUGV and LUGV, employing global features extraction to facilitate loop closure detection. When a loop closure is detected, it activates CSLAM to merge the maps generated by individual xUGVs. After map fusion, CSLAM continuously checks for intra-map closures within the unified map, refining the poses of the single agents and the associated MapPoints accordingly.

Crucially, the transformation matrix derived from visual SLAM plays an integral role in adjusting the pose of the laser point cloud gathered by the individual xUGVs. This fine integration and refinement process ultimately results in the system producing correct poses of individual agents, coordinated and refined maps, and finely tuned laser point clouds, enhancing the overall consistency and accuracy of the integrated system.

Let's take an in-depth look at the transformation process of laser point clouds. In order to correct the laser point cloud data, we need to transform it from the laser sensor's coordinate frame to the camera's coordinate frame, then apply the CSLAM loop closure correction matrix, and finally transform it back to the laser sensor's coordinate frame.

For each point p_{LiDAR} in the point cloud, use the extrinsic matrix T_{CL} to transform it to the camera's coordinate frame:

$$p_{Camera} = T_{CL} \times p_{LiDAR} \tag{1}$$

where T_{CL} is the fixed transformation relationship between the camera and the laser sensor. Then apply the loop closure correction matrix T:

$$p'_{Camera} = T \times p_{Camera} \tag{2}$$

where p'_{Camera} is the corrected point cloud in the camera coordinate and T is the CSLAM loop closure correction matrix.

$$T = \begin{bmatrix} R & \frac{t}{s} \\ 0 & 1 \end{bmatrix}^T \tag{3}$$

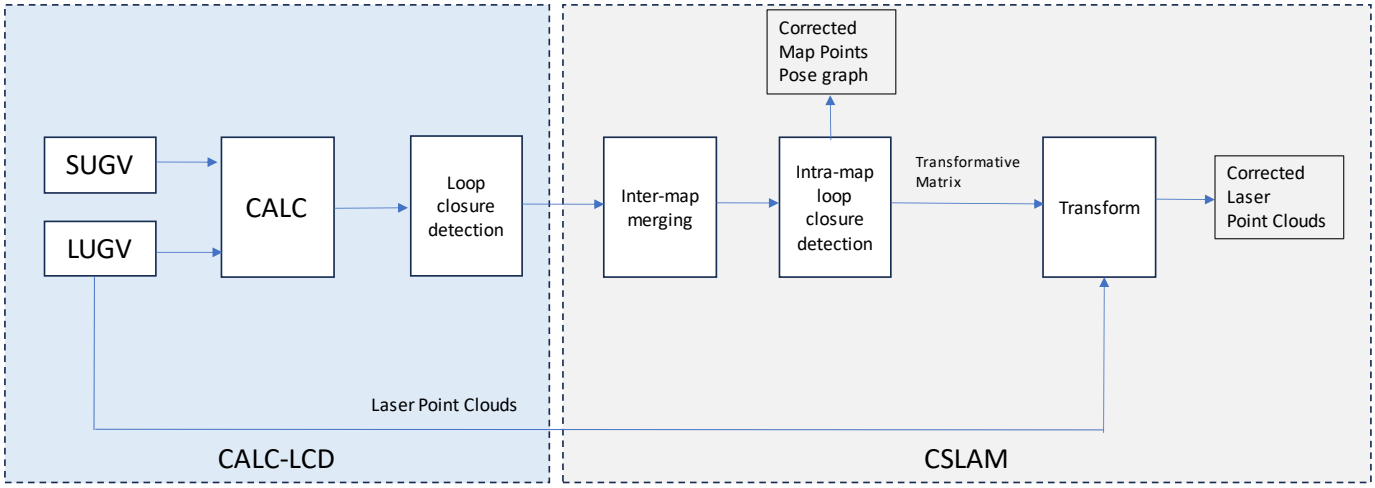

Fig. 2. The integration of CALC-LCD and CSLAM

Finally, use the inverse of the extrinsic matrix T_{CL}^{-1} to transform the point back to the LiDAR's coordinate frame:

$$p'_{LiDAR} = T_{CL}^{-1} \times p'_{Camera} \tag{4}$$

In equation (3), the top-left 3×3 block is the rotation matrix R, which is a matrix used to perform a rotation operation in a three-dimensional space. The top-right 3×1 block is the translation vector t divided by the scale s, where t is a component of a transformation that represents the shift of a point in three-dimensional space and s is used to represent a uniform scaling factor applied to the 3D points. The bottom row is $[0\ 0\ 0\ 1]$.

The rotation matrix R is a 3×3 orthogonal matrix with the property that its transpose is also its inverse:

$$R^T = R^{-1} \tag{5}$$

$$R^T R = I \tag{6}$$

where R^T is the transpose of the rotation matrix, I is the 3×3 identity matrix. R has the following structure, each element of R is the dot product of the rotated basis vectors and the original basis vectors.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{7}$$

The translation vector t is a three-dimensional vector:

$$t = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix} \tag{8}$$

where t_x, t_y, t_z are the shifts along the x, y, and z axes respectively.

Our methodology integrates a Histogram of Oriented Gradients (HOG) descriptor within a CSLAM framework, enhancing loop closure detection and mapping precision. Key advantages include improved robustness in diverse environments and accuracy through collaborative mapping. However, limitations include potential reduced effectiveness in low-visibility conditions and computational demands for real-time processing.

III. Evaluation

In this section, two different scenarios are tested with the Open Field and one scenario is tested with the Marketplace. We use the odometry of the xUGV as the external source of the xUGV's pose, which comes from the positioning of the single xGUV. We then apply CALC-CSLAM to optimise the pose of the xUGVs and compare it with CCM-SLAM (referred to as CSLAM in later sections). For evaluation, we use the Absolute Pose Error (APE), often referred to as absolute trajectory error, to measure the estimated trajectory of CSLAM and CALC-CSLAM against the external odometry source alone. Corresponding poses are directly compared between estimate and reference given a pose relation. Then, statistics for the whole trajectory are calculated.

1) Open Field: Based on the available data, we use the raw grey image and an external pose estimation source—specifically, the ZED camera's odometry from each single xUGV—instead of relying on CSLAM's and CALC-SLAM's internal tracking function.

Figure 3 displays the trajectories optimised by CSLAM and CALC-CSLAM using the ZED camera's odometry as an external pose source following inter-map loop closure detection and map fusion. As can be observed, the results indicate that when collaborative SLAM system incorporates the ZED camera's odometry, it generates more accurate poses than ZED odometry itself. The estimated trajectories of CSLAM and CALC-CSLAM align well with the ground truth from RTK GPS. Among them, the two xUGVs from the CALC-CSLAM group exhibits better performance.

2) Marketplace: Similar to the Open Field scene, we select two sub-trajectories of Marketplace scene, representing two

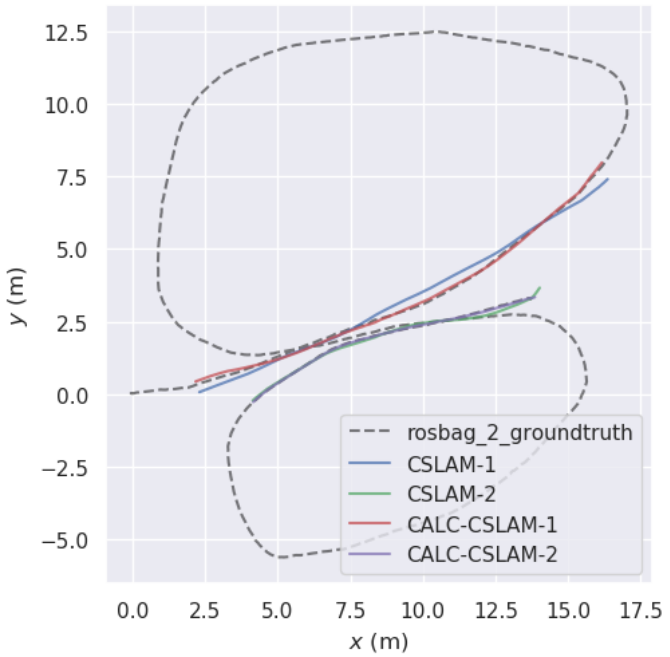

Fig. 3. Comparison of estimated trajectories in an Open Field for scenario 1. 'CSLAM-1' and 'CSLAM-2' represent the trajectories of two distinct xUGVs using CSLAM, while 'CALC-CSLAM-1' and 'CALC-CSLAM-2' represent the trajectories of two xUGVs using CALC-CSLAM. All trajectories utilize the same external pose estimation source (ZED Odometry) and are aligned with the ground truth provided by RTK GPS.

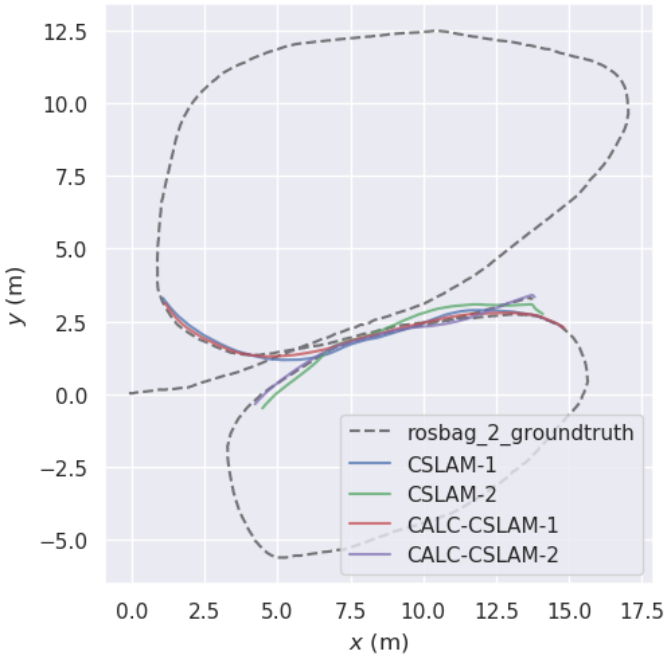

Fig. 4. Comparison of estimated trajectories in an Open Field for scenario 2. 'CSLAM-1' and 'CSLAM-2' represent the trajectories of two distinct xUGVs using CSLAM, while 'CALC-CSLAM-1' and 'CALC-CSLAM-2' represent the trajectories of two xUGVs using CALC-CSLAM. All trajectories utilize the same external pose estimation source (ZED Odometry) and are aligned with the ground truth provided by RTK GPS.

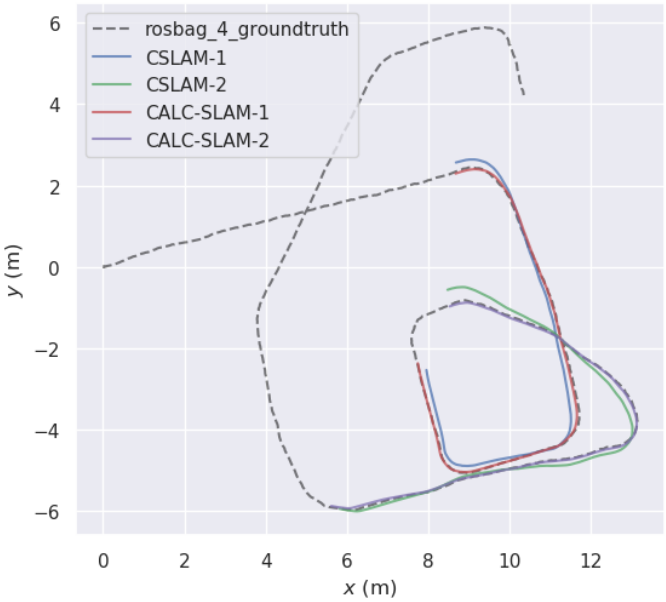

Fig. 5. Comparison of estimated trajectories in the Marketplace. 'CSLAM-1' and 'CSLAM-2' represent the trajectories of two distinct xUGVs using CSLAM, while 'CALC-CSLAM-1' and 'CALC-CSLAM-2' represent the trajectories of two xUGVs using CALC-CSLAM. All trajectories utilize the same external pose estimation source (ZED Odometry) and are aligned with the ground truth provided by RTK GPS.

single xUGVs. Each small dataset last for about 40 seconds and they have overlap in partial area which is helpful for inter-map loop closure detection. It's evident from the figure that there's an overlap between the two sub-trajectories, especially around the 8-10m mark on the East axis and around -4m on the North axis. These overlaps are beneficial, as they can be used for inter-map loop closure detection. However, due to the significant difference in view direction, the overlap around -4m on the North axis (11m on the East axis) will not generate loop closure.

Figures 3, 4, and 5 qualitatively demonstrate how Collaborative SLAM, either CSLAM or CALC-CSLAM, improves the localization of a single xUGV, while Table I provides a quantitative summary of their performance, based on the average results from five trials.

Table I gives a clear comparison of the performance metrics (RMSE, MEAN, and STD) of different SLAM techniques against an external odometry baseline. It shows that both CSLAM and CALC-CSLAM significantly reduce the RMSE (Root Mean Square Error) and MEAN of APE compared to using external odometry only, with reductions of more than 90% in all tested scenarios. This indicates that both algorithms are much more accurate in estimating the trajectory than the baseline provided by external odometry of single xUGV. CALC-CSLAM significantly enhances the performance of simultaneous localization, especially in low-contrast scenes where SLAM algorithms often struggle due to lack of distinctive features to track and map.

CALC-CSLAM consistently outperforms CSLAM in terms of RMSE and MEAN of APE across all tested scenarios, as

TABLE I
ABSOLUTE POSE ERROR (APE) STATISTICS OF THE ESTIMATED TRAJECTORY OF CALC-CSLAM COMPARED WITH CSLAM

	RMSE(m)			MEAN (m)			STD (m)		
Dataset	External odometry only	CSLAM	**CALC-CSLAM**	External odometry only	CSLAM	**CALC-CSLAM**	External odometry only	CSLAM	**CALC-CSLAM**
Open Field scenario 1	5.271	0.142	**0.054**	4.456	0.124	**0.057**	2.760	0.075	**0.025**
Open Field scenario 2	4.254	0.298	**0.113**	3.838	0.277	**0.095**	2.304	0.118	**0.043**
Marketplace	3.228	0.214	**0.081**	2.826	0.189	**0.075**	1.549	0.063	**0.032**

evident from the data in Table I. Specifically, CALC-CSLAM reduces the RMSE by about 60% compared to CSLAM, indicating a substantial improvement in trajectory estimation accuracy. Moreover, when examining the STD values, we observe that CALC-CSLAM not only minimizes errors but also maintains a stable performance across a range of environments. This consistent accuracy, observed in both Open Field and Marketplace scenarios, underscores the robustness of the CALC-CSLAM approach.

The superior performance of CALC-CSLAM can be attributed to its integration of descriptors extracted by Convolutional Neural Networks (CNNs). These descriptors encapsulate key geometric data, offering resilience against changes in illumination. The use of CNNs, trained on a diverse dataset, allows for better resistance against visual variations such as shifting lighting conditions, shadows, and occlusions. This advantage is further accentuated by the ability of CNNs to identify and leverage more distinctive features than ORB, leading to enhanced matching and alignment in collaborative SLAM scenarios. However, it's worth noting that the computational demands of CNNs render the system slower than when using the ORB feature extractor, potentially narrowing its applicability in certain situations like embedded systems.

The framework, while effective, faces challenges in low-visibility scenarios and requires significant computational resources. Future improvements could focus on enhancing visibility resilience, optimizing computational efficiency, and integrating additional sensing methods like thermal imaging or radar to overcome these limitations.

IV. CONCLUSIONS

Our research demonstrates significant improvements in multi-agent loop closure detection and collaborative mapping through the use of a collaborative SLAM framework and a convolutional neural network-based descriptor. The CALC-based CSLAM system excels in inter-map merging and optimizing the pose and map points of single xUGVs, especially in GNSS-denied environments. Replacing the ORB-BoW with the CALC descriptor has notably enhanced loop closure detection, highlighting the effectiveness of deep learning in Positioning, Navigation, and Mapping systems. Collaborative mapping from multiple agents has also refined pose estimations.

Comparative evaluations against CCM-SLAM show our system's superiority in trajectory estimation. Additionally, using visual SLAM-derived transformation matrices for LiDAR point cloud correction has improved 3D mapping and localization accuracy. Tests in varied environments like open fields and marketplaces confirm the system's robustness and adaptability. Future work will focus on leveraging corrected laser point clouds for better pose estimation

REFERENCES

[1] D. Zou, P. Tan, and W. Yu, "Collaborative visual slam for multiple agents: A brief survey," *Virtual Reality & Intelligent Hardware*, vol. 1, no. 5, pp. 461–482, 2019.

[2] I. Abaspur Kazerouni, L. Fitzgerald, G. Dooly, and D. Toal, "A survey of state-of-the-art on visual slam," *Expert Systems with Applications*, vol. 205, p. 117734, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422010156

[3] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: A survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, pp. 1–11, 2017.

[4] Z. Chekakta, A. Zenati, N. Aouf, and O. Dubois-Matra, "Robust deep learning lidar-based pose estimation for autonomous space landers," *Acta Astronautica*, vol. 201, pp. 59–74, 2022.

[5] N. Merrill and G. Huang, "Lightweight unsupervised deep loop closure," in *Proc. of Robotics: Science and Systems (RSS)*. Pittsburgh, PA: RSS Foundation, Jun 2018.

[6] H. Wang, C. Zhang, Y. Song, B. Pang, and G. Zhang, "Three-dimensional reconstruction based on visual slam of mobile robot in search and rescue disaster scenarios," *Robotica*, vol. 38, no. 2, pp. 350–373, 2020.

[7] J. Wu, R. C. Bingham, S. Ting, K. Yager, Z. J. Wood, T. Gambin, and C. M. Clark, "Multi-auv motion planning for archeological site mapping and photogrammetric reconstruction," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1250–1269, 2019.

[8] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[9] P. Schmuck and M. Chli, "Ccm-slam: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams," *Journal of Field Robotics*, vol. 36, no. 4, pp. 763–781, 2019.