City Research Online

City, University of London Institutional Repository

Citation: Chekakta, Z., Zenati, A. & Aouf, N. (2024). Enhancing Cooperative Visual SLAM with a Self-Supervised Deep Learning Model for Efficient Keypoint-Based Inter-map Loop Closure Detection. 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE), pp. 2401-2408. doi: 10.1109/case59546.2024.10711729 ISSN 2161-8070 doi: 10.1109/case59546.2024.10711729

This is the accepted version of the paper.

This version of the publication may differ from the final published version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/34026/

Link to published version: https://doi.org/10.1109/case59546.2024.10711729

Copyright: City Research Online aims to make research outputs of City, University of London available to a wider audience. Copyright and Moral Rights remain with the author(s) and/or copyright holders. URLs from City Research Online may be freely distributed and linked to.

Reuse: Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

City Research Online: http://openaccess.city.ac.uk/ publications@city.ac.uk

Enhancing Cooperative Visual SLAM with a Self-Supervised Deep Learning Model for Efficient Keypoint-Based Inter-map Loop Closure Detection*

Zakaria Chekakta, Abdelhafid Zenati and Nabil Aouf

***Abstract*— This paper introduces a self-supervised deep learning keypoint model SiLK specifically designed for loop closure detection in Cooperative Visual SLAM (Simultaneous Localization and Mapping). Firstly, the paper proposes a self-supervised learning framework that improves the robustness and accuracy of loop closure detection. Secondly, it presents the implementation of this deep learning keypoint model which replaces traditional manual feature descriptors, showcasing substantial enhancements in detecting loop closures across diverse and dynamic environments. This advancement ensures greater adaptability to variations in lighting and scene changes. Third, the paper illustrates the benefits of leveraging self-supervised learning within a cooperative visual SLAM context, where multiple agents share and fuse their local observations. This collaborative effort leads to refined pose estimations and more accurate mapping outcomes, enhancing the overall system's performance in complex settings. The effectiveness of the proposed model was rigorously tested in real-world scenarios, demonstrating its superiority in robustness and mapping precision compared to traditional methods in cooperative robotic systems, including those used for environmental monitoring and exploration tasks. The results highlight the potential of self-supervised deep learning models to revolutionize loop closure detection in visual SLAM, offering a promising avenue for future research in autonomous systems and cooperative robotics in challenging operational environments.**

I. INTRODUCTION

Positioning, Navigation, and Mapping (PN&M) are cornerstone technologies in modern automated systems and robotics, where precision and reliability are paramount. These systems are particularly crucial in environments that are dynamic and potentially unpredictable, such as urban navigation, autonomous vehicle routing, and robotic exploration. The demand for sophisticated PNM technologies is underscored in these contexts, where the ability to navigate safely and efficiently can be the difference between success and failure. As environments become increasingly complex, the role of advanced PNM systems, capable of adapting to new and unforeseen challenges, becomes indispensable. Within this landscape, the development of innovative PNM solutions that can operate in real-time and offline modes, navigating through unstructured and possibly hazardous terrains, is vital for the advancement of autonomous technologies.

The evolution of PNM technologies has been significantly influenced by advancements in computer vision, machine learning, and sensor fusion [1], [2], [3]. These technologies empower systems to perceive their environment with high fidelity, enabling the precise localization and mapping necessary for autonomous navigation. Unlike traditional PNM systems that may rely on pre-existing maps or simple sensor feedback, these advanced systems utilize a combination of sensors and computational techniques to understand and interact with their surroundings in real time. This approach not only enhances the robustness and accuracy of PNM systems but also extends their applicability to a wider range of scenarios, including those with little to no prior environmental data. Despite their potential, these advanced PNM systems face challenges such as dealing with sensor noise, dynamic obstacles, and the computational demands of processing complex environments in real time [4].

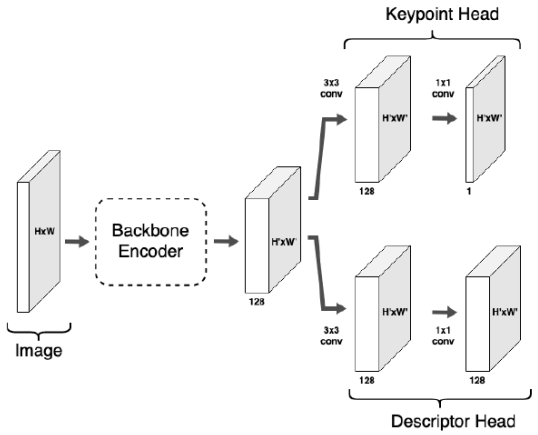


Fig. 1: Simple Learned Keypoints (SiLK): A Self-Supervised Deep Learning Approach for Keypoint Detection and Description [5]

With state-of-the-art SLAM (Simultaneous Localization and Mapping) systems have achieved high levels of accuracy and robustness, particularly in single-robot applications, and are now increasingly used in multi-robot scenarios such as search-and-rescue [6] and Explosive Ordnance Disposal missions[7]. Multi-robot SLAM enhances system robustness by sharing information among robots, increases efficiency by dividing tasks[8], and enables tasks that are impossible for a single robot. There has been a notable interest in SLAM systems designed for multiple agents, with early research focusing on range sensors like lasers and sonars for multi-agent SLAM [9]. These systems used relative measurements to estimate the collective state of all participating robots.

[1] School of Science and Technology (SST), City University of London. Northampton Square, London, EC1V 0HB, United Kingdom.

With the advancements in monocular visual SLAM, the focus has shifted to its application in multi-camera setups, improving mapping and localization by pooling spatial insights and compensating for individual limitations, leading to richer and more accurate outcomes.

As we know, Integrating inertial sensors into visual SLAM systems can mitigate failures due to suboptimal image conditions caused by poor lighting, sparse textures, or fast camera movements [10], [11], [12], [13], [14]. Techniques like VINS-Mono [15] achieve precise visual-inertial odometry by combining IMU data with visual features. Similarly, collaborative approaches such as MOARSLAM [10] and CVI-SLAM [13] enhance SLAM accuracy and robustness by sharing information between agents, while [12] demonstrates the application in swarm robotics. Ultra-Wide Band (UWB) technology further supports collaborative SLAM by allowing robots to use mutual ranging for map fusion, although its effectiveness can be diminished by obstacles that obstruct the line of sight [16].

Recent advancements also include the use of convolutional neural networks (CNNs) for depth estimation, significantly benefiting monocular SLAM systems in environments familiar to the CNN. Techniques such as CNN-SLAM [17] and D3VO [18] integrate deep learning-based depth prediction to improve pose estimation and mapping accuracy. Moreover, the development of deep learning-based feature descriptors and keypoints, as seen in DF-SLAM [19] and GCNv2 [20], showcases the potential to surpass traditional feature-based methods in terms of stability and efficiency.

This paper explores the application of collaborative robotics technology for navigation in high-risk environments. By working together, robots can enhance their capacity to evaluate, adapt to, and act in complex scenarios, thereby extending their operational range and increasing precision. Through collaborative efforts, these robots can exchange information and insights, fostering enhanced situational awareness and improved decision-making. The paper focuses on using Collaborative SLAM (CSLAM) to allow robots to collectively create accurate and coherent maps without relying on Global Navigation Satellite Systems (GNSS), and to position themselves accurately within these maps. This approach not only aids in precise map loop detection but also supports the synchronization of various data sets, facilitating the seamless integration of perceptions from individual robots into a comprehensive environmental model.

II. METHODOLOGY

A. Simple Learned Keypoints (SiLK)

The essence of SiLK is its streamlined yet adaptable approach. This method revitalizes the conventional practice of singling out distinct pixels using robust local descriptors by harnessing established contemporary methods to autonomously localize and characterize keypoints from any set of unlabeled images. Setting itself apart from traditional techniques, SiLK benefits from learned descriptors and invariances. Moreover, it simplifies the matching process by employing cosine distances and mutual nearest neighbor checks, thereby reducing the complexity and the need for extensive tuning of hyperparameters. The architecture of SiLK, featuring a straightforward backbone-plus-heads design, is notably flexible, supporting experimentation with different backbone models. Its self-supervised learning (SSL) framework allows for training on a wide array of image or video datasets without the need for annotated data. The simplicity of SiLK's training pipeline facilitates the straightforward training and assessment of various architectural designs, datasets, and hyperparameters tailored for specific tasks [5].

SiLK excels in identifying keypoints from single grayscale images, providing crucial information on both the location and description of keypoints for effective matching. Utilizing cycle consistency for descriptor learning and employing a binary classifier for the identification of distinctive keypoints at the pixel level, SiLK establishes a new benchmark in keypoint detection. To refine descriptors, SiLK processes a source image alongside a transformed variant, extracting descriptors for each point and defining transition probabilities based on descriptor similarity. This method optimizes descriptor effectiveness by enhancing cycle consistency, thus maximizing the likelihood of accurately mapping a round trip from the source to its transformed state and back.

The architecture of SiLK (illustrated in Figure 1) is inspired by the detect-and-describe strategy, similar to that of SuperPoint. Initially, an encoder backbone extracts a dense feature map from the input image, which is then utilized by two separate heads—one for keypoints and another for descriptors.

- The keypoint head processes logits to determine the probabilities of dense keypoints.
- The descriptor head generates a dense map of descriptors, which is crucial for assessing keypoint similarities.

The flexibility of the SiLK model allows for easy substitution of the backbone, accommodating various architectural preferences.

To further enhance the Loop Closure Detection (LCD) module's operational efficiency and user experience, SiLK has been adapted into the Open Neural Network Exchange (ONNX) format. This transition streamlines the process of extracting image descriptors, marking a significant advancement in the practical application of SiLK.

B. System Integration and Transformation Process

Figure 2 illustrates the integration of SiLK-LCD and CSLAM, where the SiLK module within SiLK-LCD processes images from both Small Unmanned Ground Vehicle (SUGV) and Large Unmanned Ground Vehicle (LUGV). It uses global feature extraction for loop closure detection. Upon detecting a loop closure, CSLAM is activated to merge maps created by the individual xUGVs and continues to identify and refine intra-map closures, enhancing the pose accuracy of single agents and their associated MapPoints.

The transformation matrix derived from visual SLAM is crucial for adjusting the pose of the laser point cloud collected by the xUGVs. This meticulous integration refines the agents' poses, coordinates maps, and fine-tunes laser point

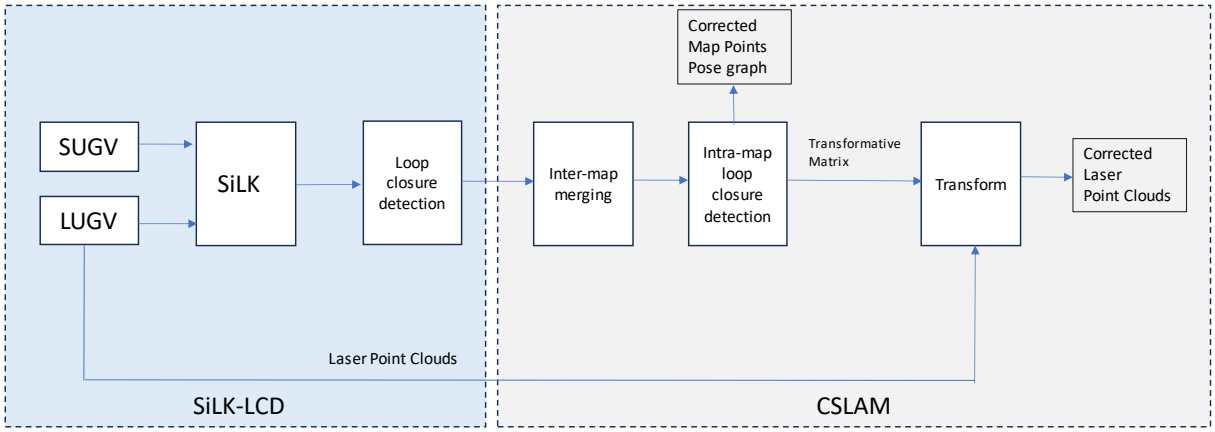


Fig. 2: The integration of SiLK-LCD and CSLAM

clouds, thereby increasing the system's overall consistency and accuracy.

C. Transformation Process for Laser Point Clouds

To adjust the laser point cloud data, it must be transformed from the laser sensor's coordinate frame to the camera's coordinate frame using the extrinsic matrix T_{CL}, corrected via the CSLAM loop closure matrix, and then transformed back to the laser sensor's frame.

For each point p_{LiDAR} in the point cloud, we transform it to the camera's coordinate frame as follows:

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

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

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

where p'_{Camera} represents the corrected point in the camera's coordinates and T is the CSLAM loop closure correction matrix:

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

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

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

In the matrix T, R is a 3×3 rotation matrix:

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

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

where R^T is the transpose of R, and I is the identity matrix. R is defined as:

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

The translation vector t is defined as:

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

where t_x, t_y, and t_z represent shifts along the respective axes.

D. Loop Closure Detection Module

As shown in Figure 3, the module aims to automatically classify a query frame produced by agent A_i into frames previously seen by agents A_j or A_k whose descriptors are accumulated and stored into dictionaries built for each feature space. These dictionaries are built online in parallel to images. Given a query image from A_i, its computed descriptors are inserted into a Search Data Structure (SDS) in the case where sufficient motion has not occurred (no loop closure is performed over recently seen images, as these will always look similar to neighbors). The SDS is then used as an input to LCD for A_j and A_k. As long as no sufficient motion has occurred, the entry relative to these frames will be updated with newly computed descriptors. In the case where sufficient motion is detected, for the descriptor in the query image, the likelihood of it belonging to each keyframe already seen by A_j and A_k is estimated. Then, the frame with the highest likelihood of the query descriptors belonging to it defines a candidate for loop closure. To find the best candidate, a fine KNN search and geometry check are used.

Once the LCD finds the loop closure candidate, it publishes the message to CSLAM as explained in section II-E. CSLAM continuously checks for loop closures in the map. Upon detecting a loop, it calculates a similarity transformation and then corrects the loop if the calculation is successful. When detecting a loop, CSLAM employs a rigorous consistency verification mechanism, spanning multiple frames, to validate these candidates. It should be noted that only when LCD finds the loop closure candidate, then it is possible for the CSLAM to continue the inter-map merging.

E. Collaborative SLAM Module

The architecture of the CCM-SLAM framework, illustrated in Figure 4, is designed to efficiently offload computationally intensive, non-real-time constrained components to a centralized ground station (referred to as the "server"), while maintaining essential modules for basic autonomy on each agent. To achieve this, each agent runs real-time VO onboard to estimate its pose and create a 3D map of its surroundings. As onboard computational resources for agents

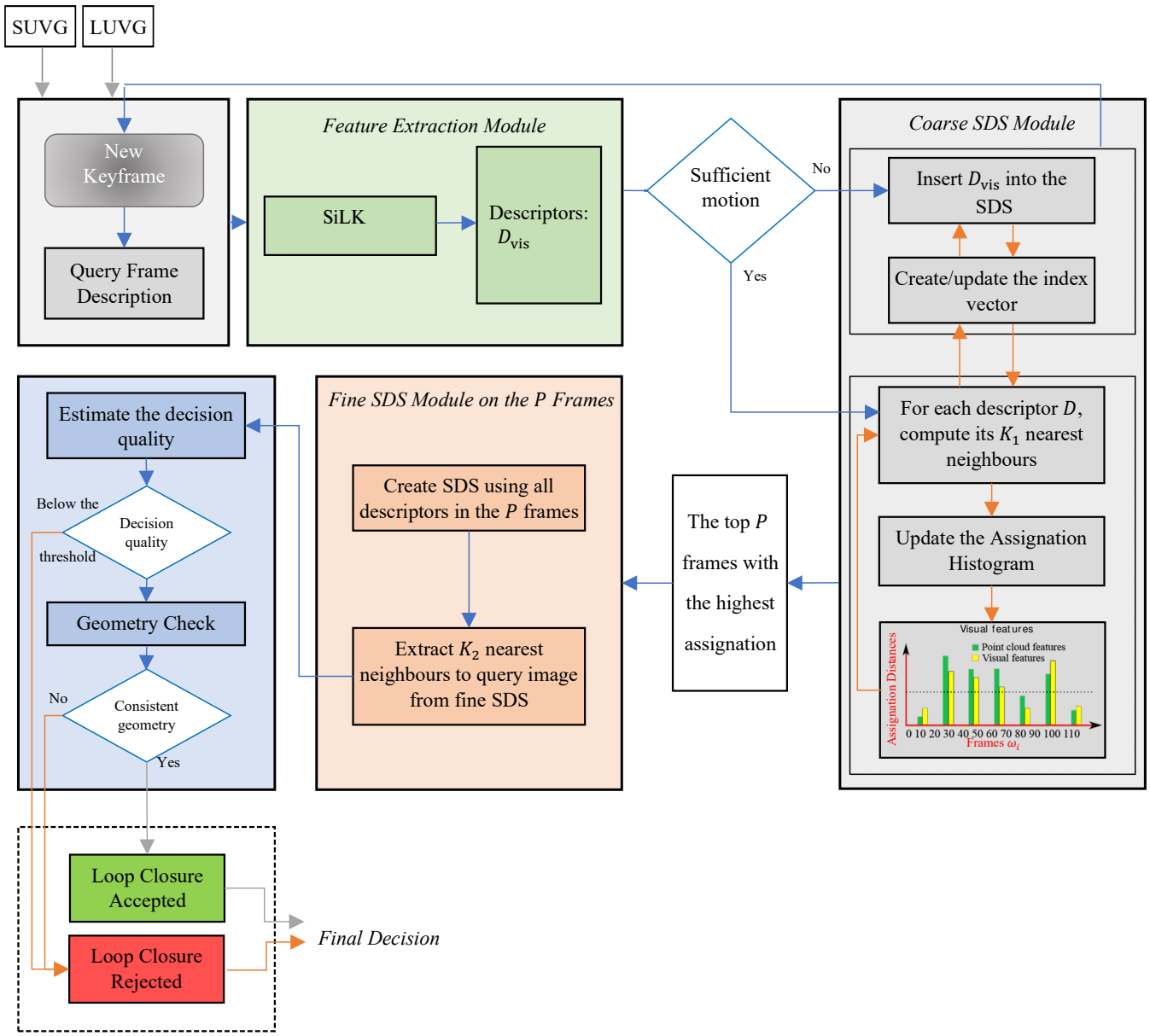


Fig. 3: The LCD module

are assumed to be more limited than those on the server, the local map generated by the onboard VO is confined to the N nearest keyframes surrounding the agents. The server acts as a record-keeper for the agents, storing and managing all experiences from every agent within the server map stack. When agents revisit a previously mapped area, the server supplies them with past experiences to enrich their local maps.

The selection of N is primarily based on the computational capacity of the agent since a larger local map increases the onboard modules' processing time. Additionally, the server runs place recognition, global optimization (bundle adjustment), and redundancy detection to eliminate highly similar or duplicate information from multiple visits to the same location by one or more agents. All maps employ a local odometry frame, with information exchanged between maps in relative coordinates from one local odometry frame to another, thus eliminating the need for a fixed global reference frame. It is important to note that CCM-SLAM does not require any prior information or configuration about the initial positions of the agents. The agents function independently until the place recognition module identifies an overlap between two maps, enabling the system to relate the relevant measurements. Bidirectional communication between the server and agents is facilitated through a wireless network, with a communication protocol capable of handling network disturbances such as delays and message loss. Both the server and agents operate a communication module responsible for map information exchange and monitoring potential errors in this exchange. In this centralized architecture, all agent

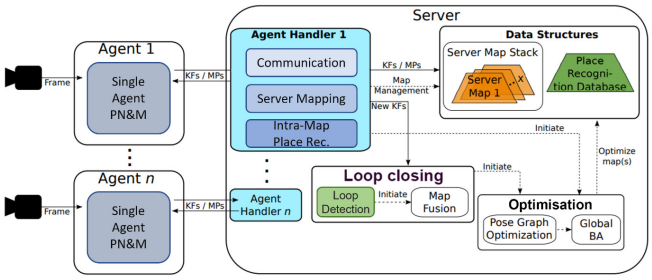


Fig. 4: Overview of the CCM-SLAM system architecture: The robotic agent operates real-time visual odometry (VO), Positioning, Navigation, and Mapping, maintaining a local map with a limited size of N, and employs a communication module for data exchange (keyframes (KF), map points (MP), and a reference KF (KF_ref) representing its current position) with the server. The server, which serves as a ground station, carries out non-time-critical and computationally demanding processes such as map management, place recognition, map fusion, and global bundle adjustment (BA). When all agents are identified to have visited the same location(s), the server's map stack holds a single, global map that integrates the experiences of all agents [21]

communication occurs via the server.

In the interaction between the Single Agent Positioning & Mapping (PN&M) module and the Server module, both KeyFrames (KFs) and MapPoints (MPs) are collaboratively shared. Notably, the pose of the agent, represented by a KeyFrame (KF), is conveyed from the single agent to the server. Subsequently, this pose undergoes optimization at the server end post the loop closure, enhancing its precision and reliability.

In this context, the CSLAM module adopts this refined pose as its primary source, bypassing the initial pose estimated by its intrinsic tracking module. However, this doesn't negate the importance of further refinements; CSLAM still actively engages in subsequent pose optimization and loop closure procedures. This means, we assign the frame's pose based on the more reliable estimation provided by the single agent and continue to refine it using the sophisticated pose optimization mechanisms inherent to the SLAM system.

This approach ensures that, during the loop closure stage, CSLAM automatically defaults to the trusted pose source associated with each frame, blending reliability with continual refinement. This amalgamation of trusted pose sources with persistent optimization and refinement procedures facilitates a more reliable and accurate representation of agent poses in the map, ensuring the integrity and robustness of the entire SLAM system.

III. SETUP

The Collaborative SLAM system is implemented in the Robot Operating System (ROS) to exchange messages. The ROS provides a run-time environment that facilitates client-server communication. Experiments are conducted to evaluate the performance of our SiLK-CSLAM system on the real datasets collected by our SUGV in two distinct scenes. During the data collection, we collect the camera images, the LiDAR Pointclouds, the RTK GPS positions, and the TF trees between them. It should be noted that the camera faces down the ground which takes about half of the image. This camera angle also helps to identify unconventional (Improvised Explosive Devices) and conventional (buried mines) explosive devices.

All results were obtained by replaying data in real-time and simulating the Small UGV and Large UGV. For these experiments, the following setup is used:

- Local Server: Intel® Core™ i5-7400 CPU @ 3.00GHz × 4, NVIDIA GeForce GTX 1060 6GB
- System: Ubuntu 20.04.6 LTS, 64-bit

Throughout all experiments, the pre-recorded datasets are processed on the same PC as the server but simulated by replaying rosbag in real-time, and communicating with the server through ROS topics, simulating a wireless network where communication takes place.

IV. EVALUATION

A. Evaluation of Estimated Trajectories

In this section, we evaluate the performance of our algorithm using data collected from the Marketplace, a closed environment specifically designed for data acquisition. The odometry data of the xUGV, derived from its positioning system, serves as the external reference for the xUGV's pose. We apply SiLK-CSLAM to optimize the pose estimation of the xUGVs and subsequently compare the results with those obtained using CCM-SLAM, hereafter referred to simply as CSLAM. For this comparative analysis, we use the Absolute Pose Error (APE), also known as absolute trajectory error. APE is used to assess the accuracy of the trajectory estimates produced by CSLAM and SiLK-CSLAM by comparing these estimates against the baseline provided by the external odometry data. This involves a direct comparison of corresponding poses between the estimated trajectory and the reference trajectory, based on their pose relation. Finally, we compile statistics for the entire trajectory to provide a comprehensive evaluation of the system's performance

Some of the detected loop closures and the corresponding matched images using the SiLK-LCD algorithm are shown in Figs. 5,6,7. The matched feature points are illustrated by the red circle marker and green plus marker in the left and right images, respectively. The corresponding feature points are connected by the yellow lines.

Figure 8 showcases trajectories optimized by CSLAM and SiLK-CSLAM, using the ZED camera's odometry as an external pose source in the Marketplace. It is evident that the estimated trajectories from both CSLAM and SiLK-CSLAM align closely with the ground truth provided by RTK GPS. The results reveal that incorporating the Zed camera's odometry into either CSLAM or SiLK-CSLAM yields poses with higher accuracy than using the Zed odometry alone, and SiLK-CSLAM demonstrates superior performance.

Table I provides a detailed comparison of performance metrics (RMSE, MEAN, and STD) for various SLAM

	RMSE(m)			MEAN (m)			STD (m)		
Dataset	External odometry only	CALC-CSLAM [22]	**SiLK-CSLAM**	External odometry only	CALC-CSLAM [22]	**SiLK-CSLAM**	External odometry only	CALC-CSLAM [22]	**SiLK-CSLAM**
Marketplace	3.228	0.081	**0.078**	2.826	0.075	**0.071**	1.549	0.032	**0.029**

TABLE I: Absolute Pose Error (APE) statistics of the estimated trajectory of SiLK-CSLAM compared with CSLAM

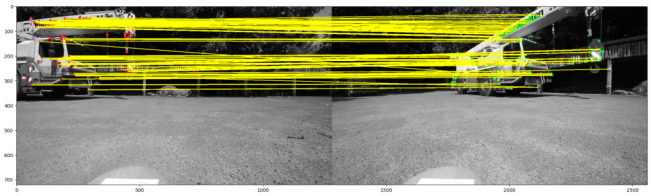


Fig. 5: Corresponded images and matched feature points using the SiLK-LCD algorithm (images: 500 vs 4188)

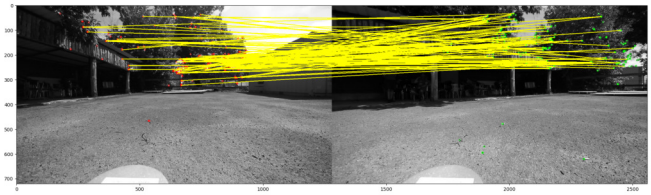


Fig. 6: Corresponded images and matched feature points using the SiLK-LCD algorithm (images: 1230 vs 2400)

techniques against an external odometry baseline. The data demonstrate that both CSLAM and SiLK-CSLAM substantially lower the RMSE (Root Mean Square Error) and MEAN of the Absolute Pose Error (APE) in comparison to solely using external odometry, achieving reductions across all tested scenarios. This substantial improvement underscores the superior accuracy of both algorithms in trajectory estimation compared to the baseline odometry provided by a single xUGV. Notably, SiLK-CSLAM shows enhanced performance in simultaneous localization, particularly in environments with low contrast, where traditional SLAM algorithms typically face difficulties due to the absence of distinct features for tracking and mapping. SiLK-CSLAM achieves better performance than CSLAM in terms of RMSE and MEAN of APE in all evaluated scenarios, as detailed in Table I. Specifically, SiLK-CSLAM shows a reduction in RMSE compared to CSLAM, which marks a significant enhancement in trajectory estimation accuracy. Additionally, when considering the STD values, SiLK-CSLAM not only reduces errors but also demonstrates stable performance across diverse environments.

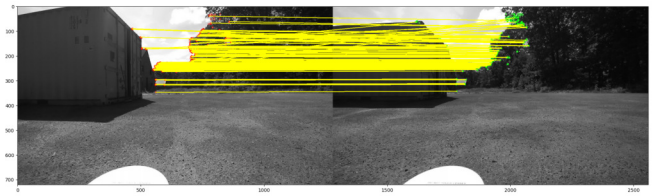


Fig. 7: Corresponded images and matched feature points using the SiLK-LCD algorithm (images: 1642 vs 2909)

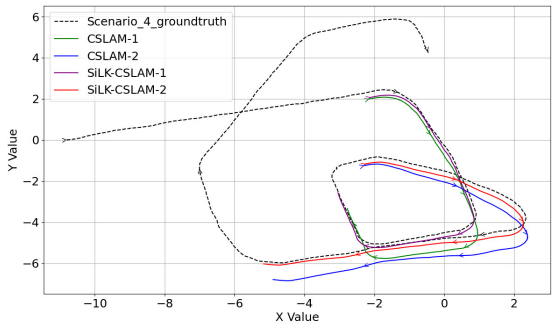
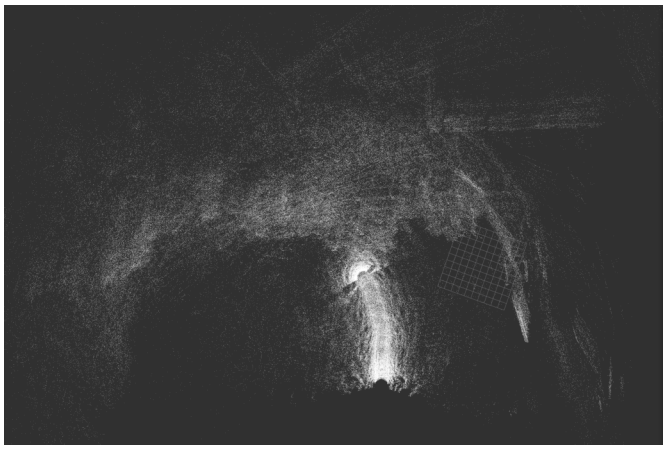
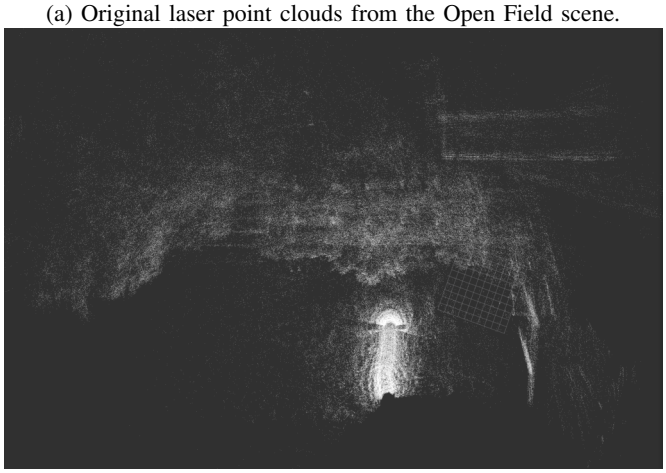


Fig. 8: Comparison of estimated trajectories in the Marketplace. 'CSLAM-1' and 'CSLAM-2' represent the trajectories of two distinct xUGVs using CSLAM, while 'SiLK-CSLAM-1' and 'SiLK-CSLAM-2' represent the trajectories of two xUGVs using SiLK-CSLAM. All trajectories utilize the same external pose estimation source (ZED Odometry) and are aligned with the ground truth provided by RTK GPS.

The enhanced performance of SiLK-CSLAM is largely due to its use of descriptors generated by the SiLK model. These descriptors capture essential geometric information, making the system robust against variations in lighting conditions. SiLK's training on a varied dataset improves its ability to handle visual changes, such as fluctuations in light, shadows, and occlusions. However, it is important to note that the computational requirements of the SiLK model make the system slower compared to using traditional ORB feature extractors. This increase in computational load may limit its use in scenarios requiring quick processing, such as in embedded systems.



(a) Original laser point clouds from the Open Field scene.



(b) Corrected laser point clouds from the Open Field scene.

Fig. 9: Laser point clouds correction.

B. LiDAR PointClouds Correction

Figure 9 shows the laser point clouds corrected by transformative matrix obtained from visual SLAM, i.e. SiLK-CSLAM. The white dots represent the laser points, with the grid serving as the fixed global frame used for reference. The central white cloud, with the brightest dots, depicts the trajectory of the xUGV, representing the foreground grass in front of the xUGV. The upper half of the clouds represent the trees in the far distance.

The Time Slot 1 of the Open Field scene is selected as the source of the laser point clouds, which encompass the accumulative *PointCloud2* messages over the selected time. The transformation matrix acquired from the SiLK-CSLAM system is applied to the laser point clouds (see Figure 9a), yielding the corrected point clouds as depicted in Figure 9b. In comparison to the original point clouds, the corrected point clouds are better aligned, as evidenced by the xUGV's trajectory and the lines in the top right corner. However, it is observable that the corrected point clouds still exhibit some drift, potentially due to the unreliable *tf* tree transform between the laser sensor and the global reference frame map.

To effectively evaluate the quality of this correction, we propose a method grounded in contrasting the alignment of point cloud sequences before and after the correction. In the preparation phase, two-point cloud sequences, namely Sequence A (before correction) and Sequence B (after correction), are acquired. To evaluate the alignment in each sequence, the degree of alignment between every two consecutive point clouds is computed. Various metrics, such as the mean squared error from the Iterative Closest Point (ICP) algorithm, are employed for this purpose.

	Point Cloud numbers	Total Errors (m)	Avg errors (m)
Sequence A (Before correction)	130	54.9	0.422
Sequence B (After correction)	130	**34.2**	**0.276**

TABLE II: Point cloud alignment error using Iterative Closest Point (ICP) registration for consecutive clouds.

From Table II, it's evident that "Sequence B (After correction)" has both a lower total and average alignment error compared to "Sequence A (Before correction)", despite both sequences having the same number of point clouds. This suggests that the transformational correction applied by visual SiLK-CSLAM was effective in improving the alignment of the related laser point clouds.

V. CONCLUSIONS

The exploration and implementation of collaborative SLAM with a Self-Supervised Deep Learning Keypoint-Based model for inter-map loop closure detection have produced significant advancements in the domain of multi-agent loop closure detection and collaborative mapping. The devised SiLK-based triggering mechanism for CSLAM has proven effective in inter-map merging and optimization of single xUGV pose and map points, paving the way for more accurate and reliable mapping solutions, especially in GNSS-denied environments.

Comparative evaluation of estimated trajectories against CCM-SLAM in both qualitative and quantitative ways provides concrete data to support the effectiveness of the system. Furthermore, the correction of LiDAR point clouds using the transformation matrix from visual SLAM has opened avenues for more accurate 3D mapping and localization. The comparative analysis has shown the robustness and adaptability of the system. In future work, we plan to explore the use of corrected laser point clouds for enhanced 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] N. Merrill and G. Huang, "Lightweight unsupervised deep loop closure," in *Proc. of Robotics: Science and Systems (RSS)*. Pittsburgh, PA: RSS Foundation, Jun 2018.

[5] P. Gleize, W. Wang, and M. Feiszli, "Silk: Simple learned keypoints," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 22 499–22 508.

[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] E. Ghisoni, S. Govindaraj, A. M. C. Faulí, G. De Cubber, F. Polisano, N. Aouf, D. Rondao, Z. Chekakta, and B. de Waard, "Multi-agent system and ai for explosive ordnance disposal," *CEIA HUMANITARIAN CLEARANCE TEAMWORK*, p. 26, 2023.

[8] Z. Chekakta, N. Aouf, S. Govindaraj, F. Polisano, and G. De Cubber, "Towards learning-based distributed task allocation approach for multi-robot system," in *2024 10th International Conference on Automation, Robotics and Applications (ICARA)*. IEEE, 2024, pp. 34–39.

[9] 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.

[10] J. G. Morrison, D. Gálvez-López, and G. Sibley, "Moarslam: Multiple operator augmented rslam," in *Distributed Autonomous Robotic Systems: The 12th International Symposium*. Springer, 2016, pp. 119–132.

[11] 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.

[12] G. Loianno, Y. Mulgaonkar, C. Brunner, D. Ahuja, A. Ramanandan, M. Chari, S. Diaz, and V. Kumar, "A swarm of flying smartphones," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1681–1688.

[13] M. Karrer, P. Schmuck, and M. Chli, "Cvi-slam—collaborative visual-inertial slam," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2762–2769, 2018.

[14] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular slam with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[15] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[16] Y. Cao and G. Beltrame, "Vir-slam: Visual, inertial, and ranging slam for single and multi-robot systems," *Autonomous Robots*, vol. 45, pp. 905–917, 2021.

[17] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6243–6252.

[18] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, "D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1281–1292.

[19] R. Kang, J. Shi, X. Li, Y. Liu, and X. Liu, "Df-slam: A deep-learning enhanced visual slam system based on deep local features," *arXiv preprint arXiv:1901.07223*, 2019.

[20] J. Tang, L. Ericson, J. Folkesson, and P. Jensfelt, "Gcnv2: Efficient correspondence prediction for real-time slam," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3505–3512, 2019.

[21] 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.

[22] Z. Zhu, Z. Chekakta, and N. Aouf, "Collaborative slam with convolutional neural network-based descriptor for inter-map loop closure detection," in *2024 10th International Conference on Automation, Robotics and Applications (ICARA)*. IEEE, 2024, pp. 352–357.