# JORB-SLAM: A Jointly optimized Multi-Robot Visual SLAM

Kaustav Chakraborty[1], Martin Deegan[2,3], Purva Kulkarni[4], Christine Searle[1], and Yuanxin Zhong[5]

[1]Institute for Robotics, University of Michigan
[2]Department of Computer Science, University of Michigan
[3]Department of Mathematics, University of Michigan
[4]Department of Electrical and Computer Engineering, University of Michigan
[5]Department of Mechanical Engineering, University of Michigan

***Abstract*—We present a new implementation of collaborative ORB-SLAM, a feature based SLAM system that operates real-time, small and large, as well as in both indoor and outdoor environments. We enhance the open-source ORB-SLAM2 implementation to use data from multiple agents. A multi-agent implementation gives us the advantages of collecting data more quickly and covering a greater area. It also adds the challenge of merging two maps online at loop closure points, thus resulting in an increased area of coverage within a shorter duration of time. Additionally, we implement a robot-to-robot loop closure mechanism through the use of APRIL tags on each of the agents to make the robots easily detectable by each other in order to add additional constraints.** [1] [2]

***Index terms*— *Localization, mapping, factor-graphs, MAP estimation, data association, SLAM, multi-agent, KITTI dataset*.**

## I. Introduction

Simultaneous localization and mapping (SLAM) algorithms provide a way for robots to both build a map of an unknown environment and locate themselves on the map at the same time. There are many variations on the original implementation for different motion and sensor models and different resource constraints. One of the most effective and clean implementations for vision based SLAM is ORB-SLAM2 [3]. ORB-SLAM2, which is built upon its predecessor ORB-SLAM [2], provides support for monocular, stereo, and RGB-D cameras. A limitation of ORB-SLAM2 is that it restricts itself to a single agent. A multi-agent system has two main advantages. Firstly, it can explore and map a greater area in the same amount of time. Secondly, with the addition of an agent recognition system, it can perform loop closures from recognition of other agents as well as map points. To address this limitation and exploit these benefits, we run ORB-SLAM2 simultaneously on two client agents and fuse the result in a server agent.

A vast majority of SLAM solutions consider a single robot in a static environment, using either sparse 2D/3D feature points or dense 2D laser range-finder data. Our work addresses the less well-studied problem of a multi-robot visual SLAM, motivated by the fact that multiple robots can complete exploration and mapping tasks in less time than a single robot can. We start from the official implementation of the existing ORB-SLAM. At the inception of the project, we began by splitting the currently existing KITTI dataset into two adjustably-sized and slightly overlapping portions, enabling the multiple robots to start their traversal at two different points, collecting information in their environment. A centralized communication system is implemented such that after completion, the subsystems (ORB-SLAM instances) send their local mapping information to a centralized server thread to perform map fusion and global optimization. This strategy mimics clients performing the front-end operation of abstracting information from the sensor models and gathering data from their local environment, leaving the server to infer abstracted data by implementing various optimization strategies on the existing information, eliminate redundancies during map fusion, and publish updates back to the client. This way, the server fuses the data gathered by both the robots.

Therefore, we present multi-agent SLAM system **JORB-SLAM**[3] in this paper. The main contributions of this work are:

1) Implementing a multi-thread based centralized multi-agent SLAM system with ORB-SLAM instances as client and a graph-based global optimizer backbone as the server.
2) Adding an additional constraint between poses when the robot travels to the same space but not recognizing the loop closure, through the addition of April tag detection [6], enabling the use of the pose of tags to enhance the graph optimization for robust SLAM.
3) Creating our own dataset to validate and evaluate the implementation of our April-tag enhancement. Tools, including open source code, are also provided.

Section II details the background of JORB-SLAM. The math models of the two core JORB-SLAM modules are discussed in Section III and Section IV. To validate our system, we use two datasets to test the robustness and performance of

[1]The code of this work has been open-sourced at https://github.com/um-mobrob-t12-w19/ORB_SLAM2. Also check our webpage for further details and a video demonstration.

[2]We also provide a demonstation video both at our website and YouTube

[3]Jointly optimized ORB-SLAM

our system. Data collection is described in Section V and the validation results are presented in Section VI.

## II. Background

### A. ORB-SLAM2

Visual SLAM using only images as external information estimates the robot position while building the environment map. The same problem has also been solved by using laser or sonar sensors to build a 2D map in a small dynamic environment. However, in a dynamic, wide ranging and complex environment, there are still problems to be solved. The use of computer vision techniques in visual SLAM, such as feature detection, characterization, feature matching, image recognition and recovery,still have much room for improvement.

During the development of vision-based SLAM methods, much exciting work has come around including ORB-SLAM and LSD-SLAM [4], which provide accurate SLAM results and real-time performance. ORB-SLAM is a versatile feature point based SLAM system and it works in both indoor environments and large-scale outdoor environments. ORB-SLAM is robust against large movement between frames and it is equipped with loop closure and relocalization abilities, which makes it suitable for real world applications.

The newer version of ORB-SLAM, ORB-SLAM2, is comprised of several parallel modules: tracking, local mapping, loop closing and place recognition. It leverages an ORB feature extractor [5] to provide feature points for frame matching and uses the bag of words method [1] for place recognition. Bundle adjustment is also performed in local mapping as part of the SLAM backend. In our work, we build our system based on the feature based ORB-SLAM structure.

### B. Multi-agent SLAM

Using a multi-agent system has a number of advantages. Firstly, it is able to fuse data from multiple robots into a single map. Secondly, it inherits the bounded-time, bounded-memory properties of a single robot SLAM algorithm (CPU and memory time do not increase with the path length). Both the multi-agent and single agent SLAM tasks consist of estimating a world state, but in the multi-agent case the world state is defined as a map of a whole environment and platform positions, including their relative orientation. These relative orientations are important because they are used to combine the two maps built with data from the two observers. There are other approaches that require a prior knowledge about the platforms orientations. These approaches try to estimate the orientations in real time. In both cases, rules are provided to correct the local world state of a single observer using data from another observer. The Figure 1 shows a high level representation of a typical multi-agent SLAM method.

- The *Platform1, ..., PlatformN* components represent executors of individual SLAM algorithm instances. They can be implemented as separate computer units or threads from a mainframe etc.
- The *Merger* block updates data from a corresponding *Platform* with external data from another *Platform-s*.
- The *Common Merger* collects all estimated local world states and combines them to produce the output world state and provide data for each local Platform to update its state.

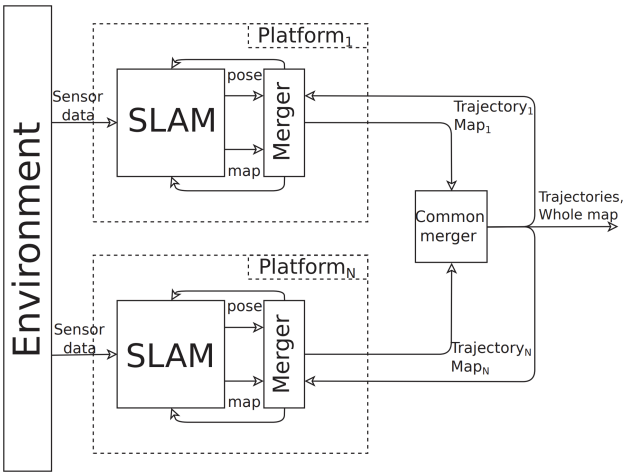


Fig. 1. Component diagram of a centralized Multi-Agent SLAM system

The challenging problem in multi-agent SLAM is merging the maps built by each separate platform. Every multi-agent SLAM method either proposes its own algorithm of merging or requires special assumptions for the environment or the world state. There are two major categories of multi-robot systems, centralized and distributed. Centralized systems have a ”server” to accept individual intermediate SLAM results and offer global map construction, while in distributed systems, each robot communicates with the others and transfers essential information to reinforce its own mapping process. There is also the choice of the interaction between the maps: a map can exist as one copy shared between all platforms, or each agent can keep and update its own representation of a map. For the latter case, the result of a merging can be presented as a common map that is identical for each platform, or every existing map can be updated independently.

Another problem is the choice of representation of the map. Map structure strongly influences merging algorithms and the optimizer implementation. Some common representations of a map are listed below:

- A graph representation of a map where each vertex stores a unique position of a platform and edges present the transaction between vertices.
- Landmarks with their co-variance matrix. Landmarks may be calculated using captured scans or may be located in the environment manually.
- A certainty grid that stores the probability for each cell to be occupied.

At the same time the individual pose updating is a critical challenge for multi-agent SLAM because observers could meet in the real world but not in their own maps. This meeting initiates a pose correction for every platform. These

corrections could happen as a jump, instantly teleporting a platform to the best found position, or through spreading the correction over several previous poses, but in the last case it is required to track several poses, excluding the last one.

A famous multi-agent SLAM system based on ORB-SLAM is Collaborative ORB-SLAM (CORB-SLAM [9]). CORB-SLAM is a centralized multi-agent system consisting of multiple extended ORB-SLAM2 instances with memory management modules. The central server of CORB-SLAM detects the overlaps of multiple local maps using Bag of Words and fuses these maps with PnP solver. To simplify the multi-agent system structure and focus on the backend, our project uses a centralized method of multi-robot SLAM, which is similar to CORB-SLAM, based on a multi-threaded system structure, with emphasis on the map fusion method.

### C. April Tags

AprilTag [6] is a visual fiducial system, useful for a wide variety of tasks including augmented reality, robotics, and camera calibration. Targets can be created from an ordinary printer, and the AprilTag detection software computes the precise 3D position, orientation, and identity of the tags relative to the camera. The AprilTag library is implemented in C with no external dependencies. It is designed to be easily included in other applications, as well as be portable to embedded devices. The fiducial design and coding system are based on a near-optimal lexicographic coding system, and the detection software is robust to variable lighting conditions and view angles.

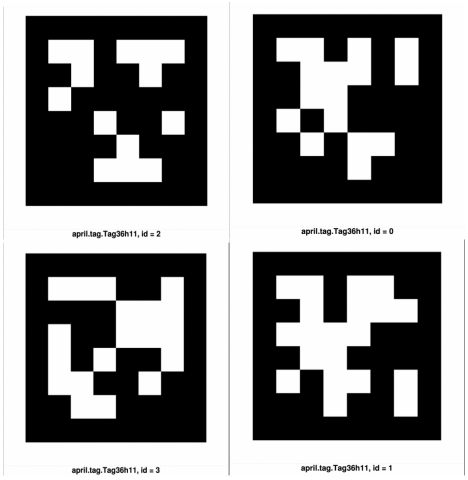


Fig. 2. 36H11 April Tag family

The April tags are the means by which the clients can detect each other. Whenever a tag is detected by a client, it signifies a loop closure. Add the constraints based on the tags and the clients' orientation is embedded as a factor in our graph, allowing us to incorporate loop closures even when the clients don't have the same scene information but are in the same location, while still using only a camera based system.

In our case we have used the April Tags of tag family 36h11. The overall process is shown in Fig. 3. The program detects the tag and then finds the homography matrix H and further the SE(3) transformation of the tag's image coordinates to the world coordinates. This transformation requires the actual dimensions of the tag and the camera intrinsics to be known.

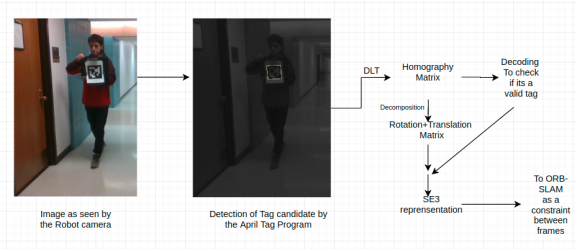


Fig. 3. April Tag detection overview

## III. Global Map Fusing

We implement an offline solution to global map fusing. This is run in five steps. Fig. 4 shows an overview of the system.

1) We first run both sequences independently. These run a largely unmodified ORB-SLAM2.
2) Upon completion, the server begins. All keyframes and map points are copied over from the individually constructed maps into the combined global map. Special care is required to copy all necessary data from each object. In addition, we maintain a map of keyframes and map points in the client maps to the keyframes and map points in the global map.
3) The covisibility graph is then reconstructed on the global map using the dictionary to look up connections in the client maps. At this point we have an inconsistent map with overlapping trajectories and features. The covisibility graph has two disjoint subgraphs.
4) Next we add each keyframe to the global loop closer to search for loop closures between sequences. This is done in four substeps.
   a) DBoW2 is used to detect possible loop closure candidates.
   b) Loop closure candidates (keyframes) from the same sequence are rejected. Each sequence should already contain self loop closures created in step 1.
   c) A similarity transform is attempted with outlier rejection for more a geometric check.
   d) Finally a loop closure edge is added between the sequences and the essential graph is optimized.
5) After processing all keyframes for loop closures, a final global bundle adjustment is performed to fix any remaining inconsistencies within the map.

## IV. Client to Client Loop Closures

We propose a new way of closing loops. Largely motivated by examples such as fig. 5, we wanted to improve multi agent mapping beyond just place recognition. Using April Tags, we can extract the relative pose of another passing agent as we are

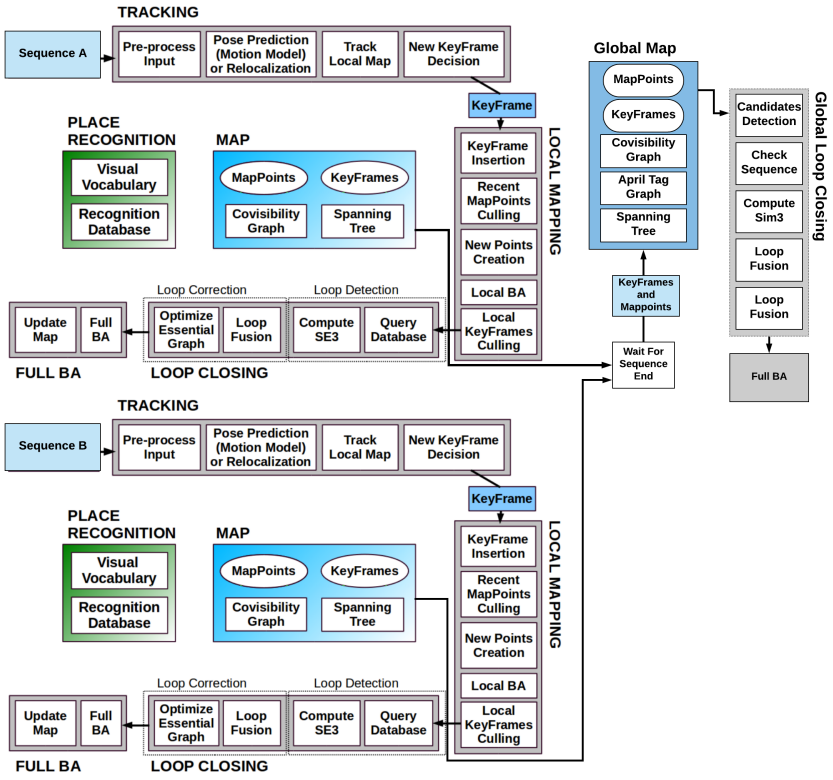


Fig. 4. JORB-SLAM System overview showing our extension to the existing ORB-SLAM2 modules.

mapping. We incorporate this information as extra constraints in the factor graph.

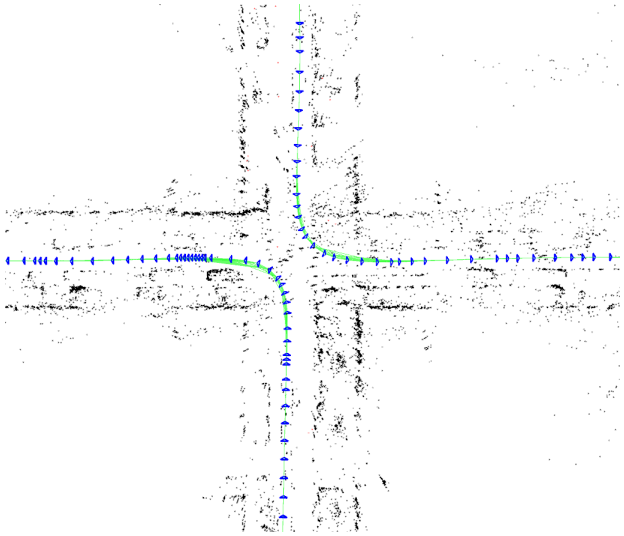

Fig. 5. A section of the KITTI 00 sequence where a loop closure should have been performed, but was not. This is a result of the inability to match ORB features under high viewpoint invariance.

### A. Detection

We begin by searching for April Tags within each frame. This is done at the same time as ORB feature extraction. With a successful detection, we extract the relative pose of the tag and store that in the frame. We modify the constraints of keyframe creation in section V-E of [2]. Keyframes are always created upon an April Tag detection. With the setup we use in section V-B one client can see another, it is highly likely that the reverse is also true. Therefore, we want to increase the number of constraints and time resolution of the trajectory during the pass. With a large number of new keyframes, we also need to modify keyframe culling of section VI-E. Keyframes flagged with an April Tag detection are automatically kept regardless of the keyframe culling request.

### B. Matching

Once both clients have finished mapping and their maps have been copied into the global map, we need to find the corresponding keyframes to add our constraints. To do this, we augment the covisibility graph to create our April Tag graph which contains the edges representing relative poses between keyframes. This allows us to quickly construct the factor graph during optimization.

Given the keyframes $\chi_A$ and $\chi_B$ from each sequence, we can query their timestamps with $t(\cdot)$. If a keyframe $X_i \in \chi_A$ has detected an April Tag, it searches keyframes from the second sequence for the closest timestamp.

$$X_i.X_{april} = \underset{X_j \in \chi_B}{\arg\min} |t(X_j) - t(X_i)| \tag{1}$$

We then check that the timestamps are close enough, otherwise the detection is rejected. $|t(X_i.X_{april}) - t(X_i)| < \delta_t$. We set $\delta_t = 0.1$ seconds.

### C. Factors

We define $\rho$ to be the objective function minimized in the full bundle adjustment of [3] (equation (4)). We then append our objective function that minimizes the error in the April Tag constraints. Let $X_A$ be a keyframe which has detected an April Tag associated with $X_B = X_A.X_{april}$. The detected April Tag gives us the relative transformation from $X_A$ to $X_B$, $\hat{X}_{AB}$. Now we define our factor:

$$r(X_i, X_j) = \frac{1}{2} \left|\left| \log\left( X_i^{-1} X_j \hat{X}_{ij} \right) \right|\right|^2 \tag{2}$$

Now we compute the Jacobians of this factor. We begin by perturbing the inputs by a small vector in $\mathfrak{se}(3)$.

$$r(X_i \exp(d), X_j) = \frac{1}{2} \left|\left| \log\left( \exp(-d) X_i^{-1} X_j \hat{X}_{ij} \right) \right|\right|^2 \tag{3}$$

Note that we can assume the error will be small in $X_i^{-1} X_j \hat{X}_{ij}$ and we can therefore replace it with $\exp(\xi_{ij})$

$$r(X_i \exp(d), X_j) = \frac{1}{2} ||\log\left(\exp(-d)\exp(\xi_{ij})\right)||^2 \tag{4}$$

$$\approx \frac{1}{2} \left|\left| \xi_{ij} - J_l^{-1}(\xi_{ij}) d \right|\right|^2 \tag{5}$$

Here $J_l$ is the left Jacobian of SE(3) [8]. Next we turn to the perturbation of $X_j$.

$$r(X_i, X_j \exp(d)) = \frac{1}{2} \left|\left| \log\left( X_i^{-1} X_j \exp(d) \hat{X}_{ij} \right) \right|\right|^2 \tag{6}$$

$$= \frac{1}{2} \left|\left| \log\left( X_i^{-1} X_j \hat{X}_{ij} \exp(\mathrm{Ad}_{\hat{X}_{ij}}^{-1} d) \right) \right|\right|^2 \tag{7}$$

$$= \frac{1}{2} \left|\left| \log\left( \exp(\xi_{ij}) \exp(\mathrm{Ad}_{\hat{X}_{ij}}^{-1} d) \right) \right|\right|^2 \tag{8}$$

$$\approx \frac{1}{2} \left|\left| \xi_{ij} + J_r^{-1}(\xi_{ij}) \mathrm{Ad}_{\hat{X}_{ij}}^{-1} d \right|\right|^2 \tag{9}$$

Here $J_r$ is the right Jacobian of SE(3) [8]. Now we add to $\rho$ to create the factor graph with Client to Client loop closures. We minimize the augmented objective function with respect to the poses of our graph

$$\{X^i, X_i\} = \underset{X^i \in \mathbb{R}^3, X_l \in \mathrm{SE}(3)}{\arg\min} \rho + \sum_j r(X_j, X_j.X_{april}) \tag{10}$$

Here, we replace $R_l$ and $t_l$ with $X_l$ in the minimization term. $\rho$ contains all the arguments to the double sum presented in equation (4) of [3].

(11)


Fig. 6. Map of the second floor of EECS building

## V. Data Preparation

The open source ORB-SLAM implementation comes with support to run on three datasets: the TUM dataset, the KITTI dataset, and the EuRoC dataset. We utilize only the KITTI dataset, modifying the provided KITTI run script to mimic two separate data streams (as from two agents) and adding a server class to coordinate the results of ORB-SLAM2 running on each agent in a client/server relationship.

### A. KITTI Dataset Reconstruction

The KITTI dataset [7] is a popular comprehensive data collection with well-defined benchmarks for tasks including object detection, object tracking, vehicle odometry and more. The dataset provides performance charts and scoring for state-of-art algorithms.

In the first phase of our testing, we utilize the greyscale stereo image pairs from odometry dataset sequence 00 of KITTI dataset. To mimic a two agent system, we split the dataset roughly in half, leaving a small amount of overlap and adjusting timestamps on the second half to mimic starting at the same time as the first half. We then feed the two halves to two ORB-SLAM2 instances.

### B. EECS Building dataset

In order to incorporate April tags and test our approach in an indoor environment, we collected a custom dataset on the second floor of the EECS building, University of Michigan. Figure 6 represents the design map of the aisles where we collect our data. The sequences were collected using Intel Realsense D435 RGB-D cameras. The sequence traverses a loop with several location loop closures and client to client loop closures. The dataset proves to be a challenge for visual odometry because of the sharp 90 degree turns.

For data preparation, we performed intrinsic calibration using MATLAB Camera Calibrator and saved the corresponding configuration files. We also synchronized the timestamps of the depth and lidar data, both at the collection step and the conversion step of the data. The tools for converting data from Intel Realsense ROS bags to a TUM formatted dataset

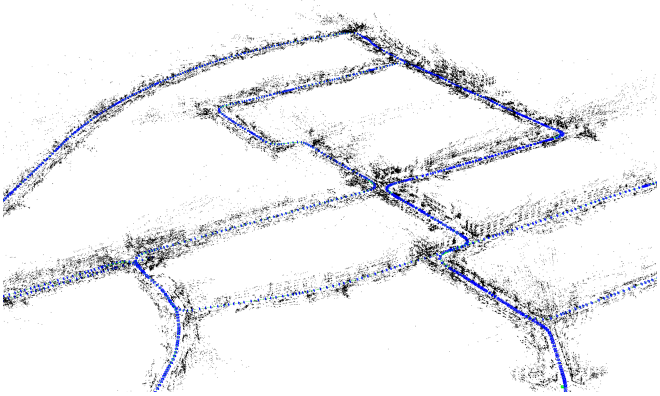

Fig. 7. Final map of our enhanced multi-agent ORB-SLAM2 system running with the KITTI dataset. Map points are shown in black, keyframes are shown in blue.

are provided in our code repository. We also provide several data sequences of the building with links published on our webpage.

## VI. Results

### A. KITTI Dataset

We first tested our enhanced multi-agent ORB-SLAM2 on the KITTI dataset [7]. Figure 7 shows a wide-view portion of the final map. Figure 8 shows a closer view of a single corridor of the dataset. In figure 8 the loop closure points (green) between keyframes (blue) can been seen.

Figure 9 shows a crossroads of two paths. Because the two paths are taken by agents moving in opposite directions, there is no ability for loop closure between the two paths. This scenario illustrates the problem we solve through April tag supported agent-to-agent recognition.

### B. EECS Dataset

The EECS dataset includes April tagged ”robots”, enabling testing of the robot-to-robot loop closure functionality. Figure 10 shows the successful robot-to-robot loop closure point at the beginning of the dataset in red lines. Green lines represent within-robot loop closures, and blue triangles represent key frames.

Figure 11 shows a wider perspective of the two agents meeting after a separation and again successfully loop closing against each other. Their positioning in this figure is analogous to the unrecognized loop closure in figure 9 - again the agents face opposite directions, but the April tag functionality enables recognition and therefore loop closure.

## VII. Conclusion and Future Directions

JORB-SLAM is a multi-agent enhancement upon ORB-SLAM system, providing greater opportunities and robustness for efficient and effective exploration. Additionally, JORB-SLAM uses April tags to detect robots and provide additional loop closure criteria. We also evaluate our approach on the KITTI dataset and a new EECS dataset, proving the feasibility of the multi-agent system and April tag detection.

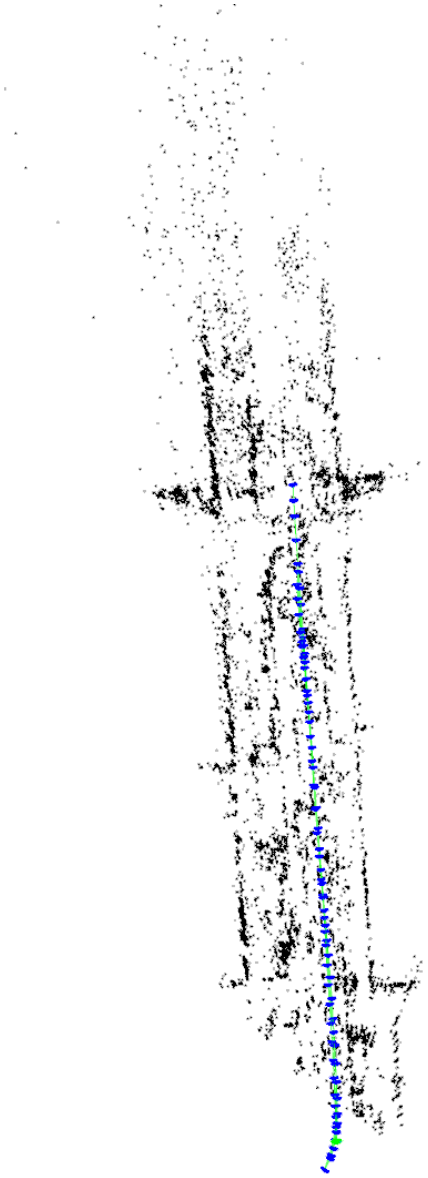

Fig. 8. Close up of a loop closure sequence in the KITTI dataset. Map points are shown in black, keyframes are shown in blue, covisibility and loop closure edges are shown in green.

While the implementation described here is limited to a software driven proof-of-concept, only minor modifications are required for implementation in a hardware system. There are various applications where the concept of our JORB-SLAM can be put to great use. One of the core application fields is in observation and monitoring systems of marine applications. The process of navigation in open sea space is very challenging, with fewer landmarks contributing to higher uncertainties in the agents. The cooperative framework approach could be effectively put to use in this situation.

On the other hand, there are still several problems that are left unsolved in our system and future exploration can be done on these topics.

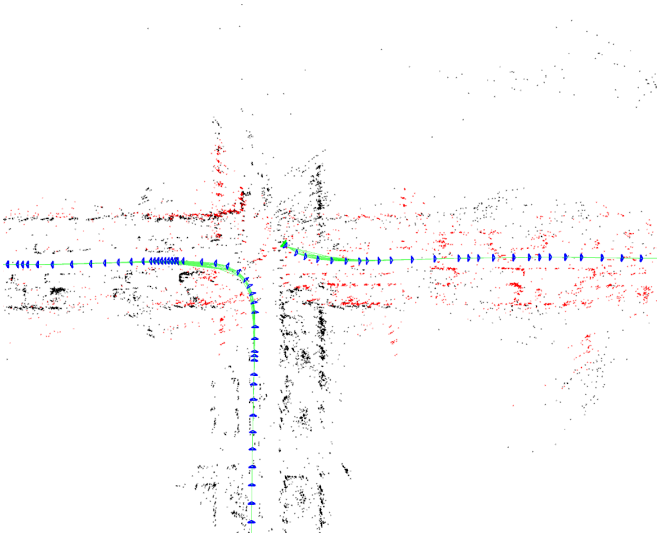

Fig. 9. KITTI dataset crossroads point. One agent moves from the "south" and turns "west", the other moves from the "north" and turns "east". The two agents travelling in opposite directions do not recognize the other's path and therefore cannot use the meeting for loop closure.

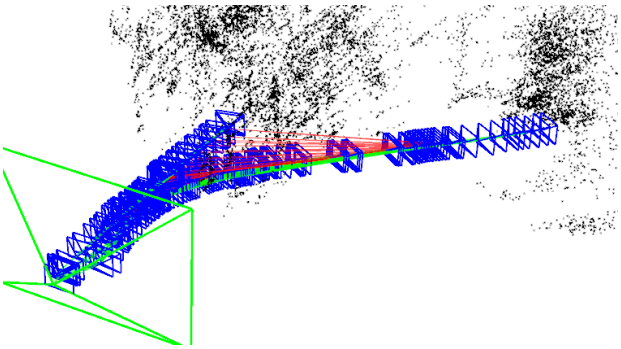

Fig. 10. Loop closure point from starting positions in the EECS dataset. Blue represents keyframes, green represents within-agent loop closure, red represents between-agents loop closure.

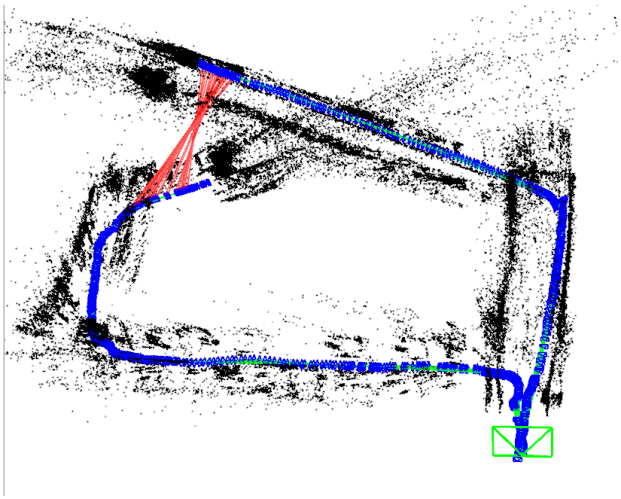

Fig. 11. Between-agents loop closure point in the EECS dataset after significant distance travelled.

*1) Robot recognition:* While extremely useful, as proven by their widespread adoption, April tags are an artificial handicap to robot recognition. A more elegant solution, a solution better able to blend in to the natural world could use learned features of the robotics agents in the system to recognize the agents instead.

*2) Data efficiency:* The navigation for autonomous agents in unknown and changing environments requires a map and efficient localization technique to enable the agents to perceive the surrounding data. However, each design varies depending on the application requirement, types of sensors, and techniques. The multi-agent collaborative ORB-SLAM implemented here is essentially a software simulation of multiple visually aided agents mapping the environment they are traversing, proving that it reduces the amount of time required to map a larger area. This idea can be further extended to the physical level through a hardware implementation of the multi-agent system. Such a system will require various additional functionality and extensions for the whole system to work for a dedicated purpose or application.

*3) Decentralized system:* The communication protocol system implemented is a centralized communication protocol. Clients communicate with each other via a centralized server, which performs the tasks of map fusion and global optimization, and also (ideally) updates the local mapping information contained in the clients. However, a major constraint in the centralized communication protocol lies in the fact that we optimistically assume a fault-free perpetually functioning server impervious to failure on accidents. This issue could be mitigated through the implementation of a distributed communication system having multiple servers, each capable of covering up or accounting for failures in the other servers. Additionally, an ad-hoc communication protocol could be implemented amongst the clients in a GPS-denied environment.

## VIII. Acknowledgement

We would like to thank Dr. Maani Ghaffari Jadidi for supporting this project and providing us valuable feedback and suggestions on the project implementation.

## References

[1] Glvez-Lpez, Dorian, and Juan D. Tardos. "Bags of binary words for fast place recognition in image sequences." IEEE Transactions on Robotics 28.5 (2012): 1188-1197.

[2] Mur-Artal, Raul, Jose Maria Martinez Montiel, and Juan D. Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system." IEEE transactions on robotics 31.5 (2015): 1147-1163.

[3] Mur-Artal, Raul, and Juan D. Tards. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras." IEEE Transactions on Robotics 33.5 (2017): 1255-1262.

[4] Engel, Jakob, Thomas Schps, and Daniel Cremers. "LSD-SLAM: Large-scale direct monocular SLAM." European conference on computer vision. Springer, Cham, 2014.

[5] Rublee, Ethan, et al. "ORB: An efficient alternative to SIFT or SURF." (2011): 2564-2571.

[6] Wang, John, and Edwin Olson. "AprilTag 2: Efficient and robust fiducial detection." 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016.

[7] Geiger, Andreas, et al. "Vision meets robotics: The KITTI dataset." The International Journal of Robotics Research 32.11 (2013): 1231-1237.

[8] Barfoot, Timothy B., et al. ”Associating Uncertainty With Three-Dimensional Poses for Use in Estimation Problems” IEEE Transactions on Robotics
[9] Li, Fu, et al. ”CORB-SLAM: a collaborative visual slam system for multiple robots.” International Conference on Collaborative Computing: Networking, Applications and Worksharing. Springer, Cham, 2017.