Centralized Collaborative Visual SLAM: A Technical Overview

Nathaniel Burgdorfer
Literature Review Paper
ME 656: Autonomous Navigation for Mobile Robots, Spring 2022
Stevens Institute of Technology
nburgdor@stevens.edu

Abstract

This work provides an overview on current state-of-the-art methods in Collaborative Visual SLAM. Several frameworks have been published regarding Collaborative Visual SLAM in which systems utilize multiple mobile robots or handheld devices with passive camera sensors in order to increase the robustness, speed, and overall quality of Simultaneous Localization and Mapping systems. In this review, we focus entirely on centralized visual monocular frameworks, with methods following a sparse, indirect, keyframe-based approach to SLAM.

1. Introduction

With the increasing availability of inexpensive, lightweight camera sensors, visual SLAM has become increasing popular for many applications. As apposed to using LiDAR, visual SLAM system use a passive observation of light and measure pixel intensities to gather information from the environment. This information is extracted from images as feature descriptors of robust, highly identifiable sections of images. SLAM systems use the identification of the same features in multiple image frames in order to reason about frame locations and 3D features coordinates in the given scene. To augment a traditional visual SLAM system, multiple robots can be introduced in a collaborative architecture in order to increase the robustness and accuracy of localization and map building. Collaborative multi-robot approaches also allow for faster coverage and mapping of a given environment. Within the context of Collaborative SLAM systems, there is a choice between a centralized or decentralized architecture. In a decentralized architecture, the robot agents communicate directly with each other, sending map information and updates directly to other robots. While these types of systems can theoretically scale to a larger swarm of robots, most literature typically focuses on a centralized approach. With a central server, much of the computationally expensive tasks, such as loop closure and global map optimizations, can be offloaded to the server. This allows the agents to concentrate all computational bandwidth towards time-critical tasks, such as visual odometry. A centralized approach also allows the robot agents to offload much of the old recorded map information to the server for storage. For the remainder of this review, we will be focusing on works that follow a centralized approach. In Section 2, we will introduce several state-of-the-art frameworks in the centralized collaborative visual SLAM paradigm. In Section 3, we will give a summary of the topic, as well as provide implementation details from each of the related frameworks.

2. Related Work

Previous works in the area of multi-robot systems have either focused on the localization or mapping problems separately. These works do not fully utilize the information provided by a multi-robot system, and can only be restricted to either improving localization estimates or improving 3D reconstruction tasks.

Looking to create a system architecture that enables the sharing of information between robots for the entire SLAM task, Morrison et al. introduce MOARSLAM [5], a client-server architecture between multiple robot agents and a centralized server. This framework uses several robot agents that run full SLAM systems on-board, while providing updates to a central server to accumulate and distribute a global map of the environment.

Schmuck and Chli introduce Multi-UAV Collaborative Monocular SLAM (MCM-SLAM) [9], a robust centralized collaborative SLAM approach that looks to take full advantage of a central server with presumably much higher computational bandwidth. In contrast to the work by Morris et al. [5], MCM-SLAM looks to offload all computationally expensive tasks to the centralized server, leaving only real-time necessary tasks on the servers. This allows the robot agents to apply their potentially limited on-board resources exclusively towards real-time tasks, such as visual odometry.

Schmuck and Chli continue their previous work with CCM-SLAM [10], a more efficient real-time, multi-robot capable SLAM framework. In their previous work, their experiments were run with pre-recorded datasets which does not thoroughly test the practicality of the proposed method. Here, they look to include a more rigorous communication infrastructure, as well as introduce a redundancy detection scheme to reduce the number of keyframes stored without compromising the information and robustness.

3. Method

Most visual SLAM pipelines choose either a filter-based architecture or a keyframe-based architecture to perform continuous localization and optimize landmark mapping. The filter-based architectures employ a variation of the Extended Kalman Filter (EKF) to estimate the ego-motion of the robot as well as the locations of any landmarks in the environment. Due to this coupling of pose and landmark estimation, filter-based approaches can be rather inefficient depending on the number of landmarks. Specifically for collaborative, multi-robot systems, it is not trivial to apply an EKF approach to enable information sharing by deploying a single filter across multiple robots.

The keyframe-based architectures rely on capturing features and their corresponding 3D points from the current frame, as well as the position and orientation of the frame. This type of approach allows the system to separate pose estimation from map optimization. A recursive approach is applied to update either the camera pose with the map fixed, or optimize the map with the pose fixed. This is more applicable to a collaborative setting and tends to be a more accurate approach.

3.1. Visual Odometry

The works that are highlighted in this review utilize a keyframe-based visual odometry front-end. This is an indirect, sparse formulation of localization and mapping. Instead of directly using pixel intensities to track position and map the surroundings, these methods opt to extract features from each frame to represent the information content of each image. Frames are then selected according to the quantity and quality of the extracted features. These selected frames are known as keyframes, and the position and orientation of these keyframes, as well as the extracted features, known as 3D map points, are stored in a local map. Using this indirect, sparse formulation is favorable in a multi-robot environment where mapping information must be passed between client and server. It is also favorable considering the potentially limited resources of the robot agents in practical scenarios. The approach taken in MOARSLAM follows the work presented by Kevian et al. [2], in which they produce relative pose estimates through the tracking of FAST corner features [7], using IMU measurements to reduce scale ambiguity. MCM-SLAM and CCM-SLAM adopt the front-end visual odometry presented in ORB-SLAM [6], in which ORB features [8] are used instead of FAST corner features.

3.2. Pose-Graph Structure

The map structure for collaborative visual SLAM is taken from the graph representation found in RSLAM [4]. This representation, named Continuous Relative Representation (CRR), maps each keyframe as a node in a graph connected by edges representing the relative transformation between each keyframe. Map points can then be projected from their 3D coordinates with respect to the keyframe that originally viewed the feature, into the current keyframe using the sequence of relative transformations following the edges of the graph. This representation is advantageous as it allows flexibility in connecting new keyframes to a map that has been adjusted through some global optimization (discussed in Section 3.7); such is the case when asynchronous keyframe updates are sent to the central server while the server is optimizing a stored map after loop closure detection.

3.3. Map Management

Centralized collaborative SLAM systems typically have to manage a trade-off between maintaining local map storage and passing map information to a centralized server. In MOARSLAM, each robot is running a full SLAM system independently, while sending periodic keyframe updates to the server. The keyframes that are sent to the server are automatically fused into a global map being maintained by the server. MCM-SLAM and CCM-SLAM maintain a local map using a select number of keyframes representing the local area around the UAV. This local map contains graph connections between keyframes and their respective 3D map points. This map also includes covisibility information between keyframes. These covisibilities are represented as undirected edges connecting two keyframes if the frames share map points greater than some threshold, N. In order to maintain a small local map, pruning is performed to keep the number of keyframes in check. In most cases once the number of keyframes becomes larger than the threshold, the oldest keyframe in the local map is removed.

The first exception to this rule is when the keyframe to be removed was not received by the server. For this reason, a buffer is used to store old keyframes until they are received by the server, in which case they will be deleted. If the buffer becomes filled due to a loss of communication between the agent and the server, then the pruning algorithm must remove keyframes. In this scenario, the local map must be trimmed in a way in which information loss of the overall collaborative system is minimized. Taking into account the possibility that an agent's local map contains keyframes captured by other agents (which will be explained in the subsequent discussion), these keyframes will be removed first. The intuition is that if these keyframes were already propagated to another agent, they must already be stored on the central server. If the buffer is still full after removing keyframes from other agents, then the oldest keyframes will be removed.

On the server, a server map is stored for each robot in the system. This server map has no maximum keyframe capacity, and therefore stores all past observed keyframes. The server also performs map fusion, which will be discussed in Section 3.6. If the server maps for multiple agents are fused together, the original server map for each of those agents is replaced by the fused map. As a product of fusion, the server also stores a transformation from each agent local map to the server map associated with the agent. The two maps are initially aligned, but as a result of potential map fusions, the server must keep track of this transformation.

In the case of CCM-SLAM, the server also detects and removes some redundant keyframes. The server randomly selects a keyframe and compares the feature and map point content of this keyframe to neighboring keyframes with high covisibility. If the overlap of map points and features is over a threshold percentage, then the keyframe is removed from the server map. This allows the removal of redundant information and keeps the server map size to a minimal size without loss of information.

3.4. Communication

The communication framework for MCM-SLAM and CCM-SLAM operate in a similar manner; however, CCM-SLAM incorporates a few key assumptions and optimizations to better handle real-time communication between several mobile robots. CCM-SLAM separates the messages sent between the agents and servers into a ”new data” message and an ”update” message. When new keyframes are generated, the new keyframes as well as the corresponding map points must be propagated throughout the collaborative system. This requires rather large messages to be passed over a wireless system. Instead of always sending the entire package of information, this work looks to just pass the necessary updated information, such as keyframe and map point pose updates. Separating these two types of messages into new and updating messages allows for much less data traffic over the wireless system. With this modification, the average reported network traffic was reduced from 10 MB/s to 0.37 MB/s.

3.5. Loop Closure

MOARSLAM follows the loop closure algorithm of Gálvez-López and Tardós [1]. This algorithm is employed on both the client and the server. Essentially, a bag-of-words database is used to compare features across images for the purpose of place recognition. The ORB features are computed around the stored FAST corner features for each image. These ORB features are converted into a bag-of-words vector that is used to query the database. Potential matches are computed based on feature similarity and relative keyframe position. When enough local keyframe similarities are accumulated, loop closure is performed by computing the local transformation between the keyframes, adding an edge to the pose-graph. The server loop closure task is triggered when a robot agent uploads a keyframe to the server for matching. If a loop closure match is found, the server sends the matching keyframe and the new transformation edge to the robot. The robot can also request that the local connectivity surrounding the matched keyframe be transferred to incorporate a small section of the global map into the local agent map. MCM-SLAM and CCM-SLAM uses two different modes of detecting and propagating loop closures. Searching for loop closures happens on the server and is completed between keyframes in a single server map, named Intra-Map Place Recognition, and between different server maps, named Map Matching. Intra-Map Place Recognition uses the current keyframe to query all keyframes from the agent server map. The query is performed over the feature space of the keyframes, returning the keyframe that maps the same map points. This trajectory overlap is used to help optimize the entire trajectory and map accuracy through Bundle Adjustment discussed in Section 3.7. Map Matching searches for overlap between two server maps. If an overlap is found, this overlapping keyframe information is used in the map fusion stage discussed in Section 3.6 to combine the server maps.

3.6. Map Fusion

Map fusion is a relatively straight-forward process. Two maps with corresponding keyframes and map points are combined to create a fused server map including information from both separate maps. One server map is used as the new frame of reference, while the other is transformed to align with the reference map. The overlapping keyframes and map points are combined, and a global bundle adjustment is run to optimize the new server map.

3.7. Global Optimization

Any time the system detects loop closures or map overlaps, a global optimization step is performed. In the case of MOARSLAM, adaptive-window bundle adjustment presented by Kevian et al. [2] is used when new keyframes are added to the pose-graph. Following ORB-SLAM [6], MCM-SLAM and CCM-SLAM perform a reduced pose-graph optimization on a subgraph of highly covisible keyframes before running Global Bundle Adjustment (BA). The BA used here is the Levenberg-Marquardt algorithm implementation provided in g2o [3]. Global Bundle Adjustment is run in order to refine the accuracy of the map. Specifically, BA aims to increase map accuracy by minimizing the re-projection errors of all key frames and 3D map points. Since this is a relatively slow process (on the order of seconds), this is run entirely on the server to allow the robot to continue running the necessary real-time processes. This refinement gets propagated to the agents with periodic updates from the server.

4. Conclusion

For practical application that include small UAV robots with potentially limited resources, a centralized collaborative approach to SLAM can aid in increasing robustness and mapping accuracy. MOARSLAM introduces a framework for a client-server architecture to allow robot agents to store, fuse and share information in an environment. MCM-SLAM looks to extend, and capitalize on the centralized approach, focusing on allowing the robot agents to prioritize necessary real-time computations while off-boarding computationally expensive tasks to the central server. CCM-SLAM aims to take the initial framework developed in MCM-SLAM and extend it to experiments involving practical datasets and live experiments, with contributions involving optimizing the global map storage system and reducing the overall network traffic required to pass information throughout the system. There are clear advantages over a single entity SLAM system, with some distributed systems challenges as a result. Overall, collaborative visual SLAM is a promising direction as small form-factor computation power becomes more widely available, with areas of impact reaching from AR/VR applications to search-and-rescue scenarios.

References

[1] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012. 3

[2] N Keivan, A Patron-Perez, and G Sibley. Adaptive asynchronous conditioning for visual-inertial slam. In *International Symposium on Experimental Robotics*, 2014. 2, 3

[3] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G¡sup¿2¡/sup¿o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011. 3

[4] Christopher Mei, Gabe Sibley, Mark Cummins, Paul Newman, and Ian Reid. Rslam: A system for large-scale mapping in constant-time using stereo. *International journal of computer vision*, 94(2):198–214, 2011. 2

[5] John G Morrison, Dorian Gálvez-López, and Gabe Sibley. Moarslam: Multiple operator augmented rslam. In *Distributed autonomous robotic systems*, pages 119–132. Springer, 2016. 1

[6] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 2, 3

[7] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006. 2

[8] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. 2

[9] Patrik Schmuck and Margarita Chli. Multi-uav collaborative monocular slam. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3863–3870, 2017. 1

[10] Patrik Schmuck and Margarita Chli. Ccm-slam: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams. *Journal of Field Robotics*, 36(4):763–781, 2019. 2