·Review·

Collaborative visual SLAM for multiple agents: A brief survey

Danping ZOU[1*], Ping TAN[2], Wenxian YU[1]

1. *Shanghai Key Laboratory of Navigation and Location-Based Services, Shanghai Key Laboratory of Intelligent Sensing and Recognition, Shanghai Jiao Tong University, Shanghai* 200240, *China*

2. *School of Computing Science, Simon Fraser University, Burnaby* V5A 1S6, *Canada*

***Corresponding author,** dpzou@sjtu.edu.cn

Received: 21 July 2019 **Accepted:** 25 September 2019

Supported by Project Grant JZX7Y2-0190258055601 and National Natural Science Foundation of China (61402283).

Citation: Danping ZOU, Ping TAN, Wenxian YU. Collaborative visual SLAM for multiple agents: A brief survey. Virtual Reality & Intelligent Hardware, 2019, 1(5): 461—482
DOI: 10.1016/j.vrih.2019.09.002

Abstract This article presents a brief survey to visual simultaneous localization and mapping (SLAM) systems applied to multiple independently moving agents, such as a team of ground or aerial vehicles, a group of users holding augmented or virtual reality devices. Such visual SLAM system, name as collaborative visual SLAM, is different from a typical visual SLAM deployed on a single agent in that information is exchanged or shared among different agents to achieve better robustness, efficiency, and accuracy. We review the representative works on this topic proposed in the past ten years and describe the key components involved in designing such a system including collaborative pose estimation and mapping tasks, as well as the emerging topic of decentralized architecture. We believe this brief survey could be helpful to someone who are working on this topic or developing multi-agent applications, particularly micro-aerial vehicle swarm or collaborative augmented/virtual reality.

Keywords Visual SLAM; Multiple agent; UAV swarm; Collaborative AR/VR

1 Introduction

Visual simultaneous localization and mapping (SLAM)[1,2] is a key technique that has been receiving significant interests from computer vision, robotics, and augmented reality communities. 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[3], autonomous navigation for robotics[4]. Comparing with SLAM techniques using other sensors like LiDAR, 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.

In the last decade, visual SLAM has developed quickly and is growing as a mature technique that has been successfully applied to several commercial products such as Microsoft's Hololens headset[1], Google Project Tango tablet[2]. The top two operation systems for cellphones, Android[3] and iOS[4], are also attempting to include the visual SLAM technique as a middleware to enable augmented reality experience for users

[1] Microsoft Hololens.

[2] Google Project Tango.

[3] Google ARCore.

[4] Apple ARKit.

with consumer-grade cellphones. Visual SLAM has also been successfully applied to consumer-grade drones. For example, several models of DJI's quadrotors, such as Mavic Air, Phantom 4, are equipped with visual SLAM systems for reliable navigation in GPS-denied environments. Visual SLAM has also become increasingly popular on ground robotics, and autonomous driving cars because of its low-cost sensors and great potential on perception, though LiDAR-based SLAM dominates those areas traditionally.

Though there are plenty of methods or systems developed for visual SLAM, only a few of them focus on purposes that involve multiple agents. Those agents may be a team of moving ground vehicles, flying drones, or a group of people that use different AR devices in the same environment. In such kind of circumstances, the agents should not only work independently without keeping aware of the movement of other agents, or the 3D maps produced by other agents. For example, a flying drone in a group needs to know not only the pose of itself for trajectory control, but also the poses of other drones for collision avoidance or path planning. Three dimensional maps produced from different drones are required to be fused to speed up the mission of exploration. For multi-user augmented/virtual reality (AR/VR), the user should also know the state of other users, and different users should be put into a unified coordinate system such that virtual objects can be seamlessly displayed at different individual devices. All those issues mentioned above therefore require the visual SLAM systems in different devices working *collaboratively* with each other, so that individual systems could share common information, to solve the localization and mapping problems simultaneously for multiple agents, usually within a global coordinate system.

In this article, we investigate collaborative visual SLAM for multiple agents. We briefly review the history about visual SLAM and then provide a rough taxonomy of existing approaches to collaborative visual SLAM. After that, we discuss about two major approaches: filter-based approach and key-frame based approach, as well as the key components such as pose estimation and mapping involved in developing a collaborative visual SLAM system. Finally, we discuss about the emerging topic of decentralized architecture. To our best knowledge, this is the first survey on multi-agent SLAM that focuses on using mainly the video cameras.

2 A brief history of visual SLAM

SLAM has been being an important research topic in robotics community since the 1980s in the last century[5,6]. It is a problem that needs to be solved when a robot is moving autonomously in an unknown environment. As the environment is unknown, the map requires to be built on-the-fly. Based on the map, the location (also the orientation) of the robot is also computed and fed into the control loop of motion planning of this moving robot. Since the localization problem and the mapping problem are typically solved simultaneously, this technique is usually called Simultaneous Localization and Mapping, or SLAM for short.

In the early stage of SLAM research, most works focus on using LiDAR as the primary sensor of a SLAM system. Since the beginning of this century, as the computational power of personal computers grows significantly and the Structure-from-Motion (SfM)[7] technique becomes mature in computer vision area, cameras gradually become a favorable sensor for SLAM problem because of its low costs, light weight, and rich information for perception. Visual SLAM, namely the SLAM system using the video cameras as the major sensor, becomes increasingly popular since the first real-time system[8] came out in 2003.

In this MonoSLAM system[8], the visual measurements, or the feature points extracted on the image, are integrated into an extended Kalman filter (EKF)[9] to estimate the ego motion of the camera and the 3D

coordinates of those feature points. Since then, a large number of algorithms of visual SLAM have been proposed. Klein et al. proposed a PTAM system[10], where they adopt a pipeline similar to that of SfM[11] while putting the time-consuming bundle adjustment into a separate thread to ensure the real time performance. In comparison with EKF-based visual SLAM system, this kind of methods that incorporate optimization (bundle adjustment[12]) as the backbone are usually faster and more accurate if the same number of feature points are used. Hence this optimization-based approaches are widely used in the later visual SLAM systems, such as ORB-SLAM[13]—one of the most complete monocular/stereo visual SLAM system, CoSLAM[14]—a visual SLAM system for a group of independently moving cameras, RKSLAM[15]—a robust visual SLAM that handles fast camera motions.

A typical visual SLAM system can be roughly classified into two parts. The first part (usually referred as the front end) involves mostly image processing, such as feature extraction from images, matching or tracking those features across different video frames. The second part (the back end) is in charge of geometric computation—converting those feature tracks into camera poses and 3D coordinates based on the theory of multiple view geometry. As described previously, the geometric computation can be implemented in two ways—through a filter[8,16,17] or a nonlinear least squares optimizer[18,19]. Additionally, there are also issues about detection of revisited places to eliminate accumulated errors (loop closure)[20] or recover the failure (re-localization)[21,22].

Some recent methods operate directly on pixels without feature extraction[23,24]. This kind of methods are known as direct approach. Experimental results show that the direct approach may lead to higher accuracy if the photometric calibration of the camera is well processed[25]. There are also semi-direct approaches[26,27] that take the advantage of both sides to achieves good efficiency and accuracy simultaneously.

Classical visual SLAM methods that rely only on vision sensor—the video cameras—may easily fail in practical cases, when the camera undergoes pure rotation or small-parallax motion, or when sometimes the camera is moving in a texture less or dark scenes. It was not until the MEMS inertial measurement unit (IMU) has been integrated that visual SLAM becomes a reliable method for practical use. The inertial information (acceleration and angular rate) renders the scale observable and makes the system robust to missing or bad vision sensory inputs. Visual SLAM system aided with the IMU sensor is usually named as a visual-inertial navigation system[28–31]. Currently, most commercial products integrated the IMU sensor as an extra aiding source for visual SLAM system to achieve better robustness and accuracy.

Though numerous visual SLAM methods have been proposed in recent years, most of them focus on the application to one single platform. While realizing a visual SLAM for a single platform is challenging enough, deploying a visual SLAM to multiple platforms involves more challenges.

Applications like a team of agents, such as drones, robots, and multiple-user devices are receiving a great of interest in recent years, especially when the fifth generation (5G)[32] of mobile networks are emerging. It hence becomes an increasingly important research topic about deploying an efficient and reliable visual SLAM system for a team of agents. In the following sections we will review recent developments on collaborative visual SLAM and provide a rough taxonomy of existing approaches.

3 Existing approaches to collaborative visual SLAM

Requests for multiple agents (such as robots, AR/VR, and wearable devices) working together are becoming increasingly popular within different scenarios. The SLAM systems for multiple agents, have been receiving a great of interests since the last decade. Most works focus on using range sensors such as

laser and sonars for multi-agent SLAM in the early stage, and attempt to use relative measurements (range and bearing)[33,34] between different agents to estimate the overall state of all agents. As monocular visual SLAM is developing quickly and becomes increasingly robust, it is naturally to consider if visual SLAM could be deployed to multiple cameras, particularly a team of independently moving cameras.

Most collaborative visual SLAM systems adopt a centralized architecture, that means the systems consist of the agent-side frontends and one server-side backend. The frontends are usually responsible for the computation of the real-time states of agents that are critical for online applications. The backend in the server is usually in charge of map fusion, simplification, and optimization. Centralized architecture is easy to be implemented and deployed compared with the decentralized architecture that are investigated in several works[35,36]. Existing approaches that adopt the centralized architecture are listed in (Table 1).

Table 1 A list of existing multi-agent visual SLAM systems

Year	System	Frontends	Backend	Map	Initialization	#Agents	Agent type
2008	*PTAMM[37]	Camera pose estimation	Triangulation, Re-localization, bundle Adjustment	One global map	Two key frames by translation of one camera[10]	2	A hand-held camera & a wearable cameras
2012	CoSLAM[14]	Image acquisition	Inter/Intra-camera pose estimation, camera mapping, grouping, bundle adjustment	One global map	Structure-from-motion[38] by cameras facing toward the same place	3/4/12	Hand-held cameras
2015	*Multi-robot CoSLAM[39]	As above	As above	As above	As above	2	Cameras on Ground robots
2013	*CSfM[40]	Visual odometry	Place recognition, map merging, pose graph optimization, bundle adjustment	Multiple local maps	Initialized by two key frames for each camera[41]	2	Cameras on two quadcopters
2013	C2TAM[42]	Camera pose estimation	Triangulation, Re-localization, bundle adjustment	Multiple local maps	Initialized by EKF for each camera[43]	2	RGBD cameras mounted on laptops
2016	*Flying Smartphones[44]	Visual-inertial odometry, UKF state fusion	Map merging, refinement	Merged local maps	Filter initialization for each camera[45]	3	Smartphones mounted on flying drones
2016	*MOARSLAM[46]	Visual-inertial SLAM, Loop closure	Place recognition, key frame storage/retrieval	Maps with relative pose connection	Key-frame optimization[47]	3	Google’s Tango Phones
2018	*CCM-SLAM[35]	Visual odometry	Place recognition, loop closure, map fusion, Remove redundant key frames	Multiple local maps	Two-frame initialization[13]	3	Cameras on AscTec UAVs
2018	CVI-SLAM[36]	Visual-inertial odometry	Loop closure, map fusion, refinement	Multiple local map	Loosely-coupled visual-inertial initialization[2]	4	Different sequences on EuRoC datasets[48]

Notes: *It is an online system.

Right after the first key-frame based monocular visual SLAM system[10] emerged, there was the first attempt to extend the monocular visual SLAM to two independently moving cameras[49], one on the shoulder and one in hand as shown in (Figure 1). The PTAMM system, as an extension of PTAM, adopts two camera trackers that run simultaneously and feed key frames to the same map maker that optimizes both the 3D coordinates of the feature points and the six degree-of-freedom (6DoF) poses of the camera trajectories. The system also stores local maps and reuses them when revisiting the same places.

We can see that the extension to multiple camera system is straightforward in the key frame-based framework, simply by adding extra camera trackers. All the major components, such as camera pose estimation, feature matching, triangulation, re-localization, and bundle adjustment remain the same as those of a monocular system. It has been shown that more powerful wearable AR application had been made by incorporating two independently moving cameras in the system.

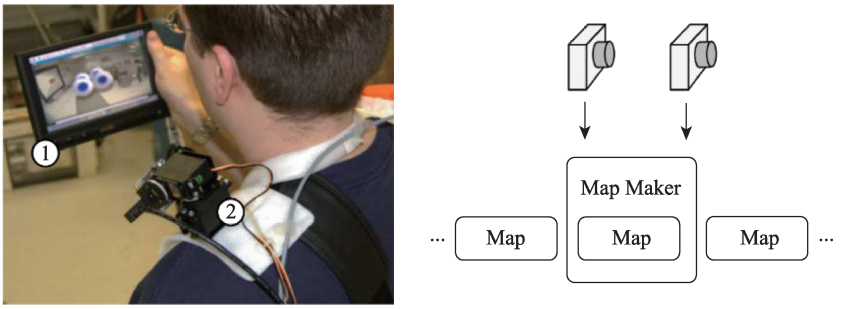

Figure 1 Left: The setting of two independently moving cameras presented in [50]. ① hand-held camera, and ② camera on the shoulder; Right: The framework of PTAMM.

Following the key-frame based framework, [14] presents a collaborative visual SLAM named as CoSLAM that supports twelve independently moving cameras. The situation becomes much more complicated than that of using only two cameras as in [49]. To fully explore the collaboration between cameras, CoSLAM[14] classifies cameras into different groups. The cameras in the same group need to have common field of views such that common feature points can be detected among those cameras. Collaboration is done within the camera group by adopting inter-camera feature matching, pose estimation, and mapping. This work has demonstrated that the system works well even when there exists a large portion of moving objects in the image, if those cameras are synchronized. The system can also estimate the 3D trajectories of the moving feature points as shown in (Figure 2), rather than treating them as outliers as in typical visual SLAM systems. This is achieved by inter-camera mapping through triangulation of 3D moving points from their corresponding feature points in different views.

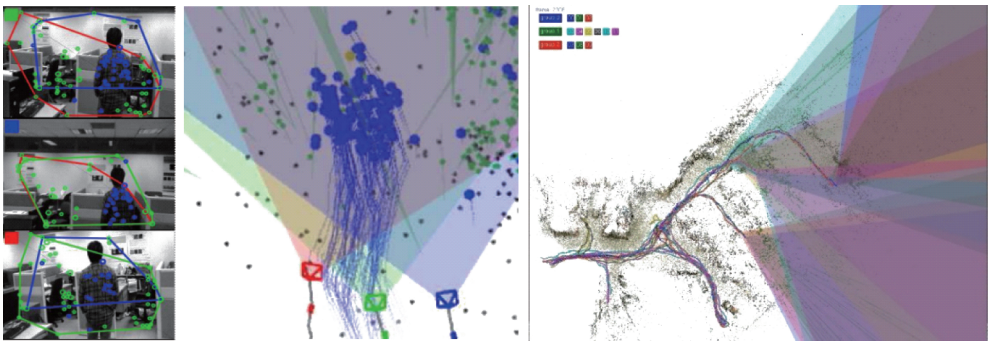

Figure 2 Left: CoSLAM[14] can estimate the 3D trajectories of moving points apart from localization and mapping. Right: Twelve cameras are classified into different groups according to their common filed of views.

The system is then deployed to two cleaning robots[39] to automatically orbit a moving target in real time. Each robot equips with a video camera sending images to the ground station. The ground station receives the images from both video cameras and runs the CoSLAM system (Figure 3). It has demonstrated that, with the help of the real time camera poses and 3D feature trajectories of the moving target, the robots can be reliably controlled to orbit the moving target, even sometimes the target moves out of the field of view.

Though achieving impressive results, those systems heavily rely on the communication capability. The communication load becomes prohibitively high when the number of cameras increases, since all the video sequences need to be sent to the ground station in real time. Another disadvantage is that the computation power on each agent is totally wasted since all the computations are done in the server—the ground station.

A straightforward approach to reducing the reliance on communication is to put some computational components back to individual agents. Each agent needs to send only intermediate results instead of video streams to the sever. Those data are usually much smaller than the raw videos such that the communication

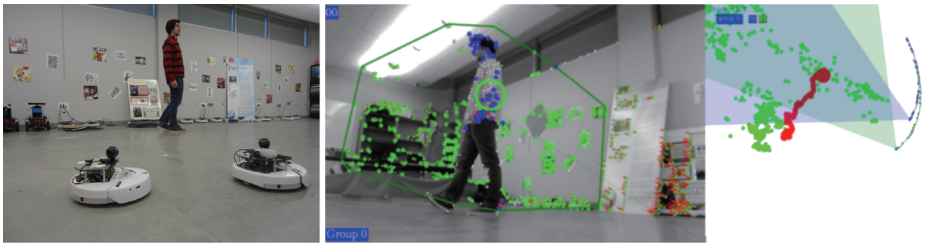

Figure 3 CoSLAM was deployed to an online system[39] that consists of two ground robots with two wireless cameras to orbit the moving target.

load will be significantly reduced. In the system presented in C2TAM[42], each agent runs the camera tracker and sends all key frame images to a cloud server. The server uses those key frame images to compute the 3D local map for each agent, and fuse those maps if the same place has been recognized. It also runs bundle adjustment to refine those maps. The agent however needs to download the updated map information from the server from time to time for camera pose estimation and re-localization.

In CSfM[40], a multi-agent visual SLAM system is developed in a similar way where each agent runs a monocular visual odometry (SLAM without loop closure) independently and sends the key frame data to a centralized sever at the ground station. The server uses the key frame data to compute the 3D local map for each agent. The key frame data however contain only the feature points on the key frame images as well as the relative poses between those key frames. Hence the communication load can be further reduced compared with that in C2TAM[42]. To obtain the absolute pose of the newly arrived key frame in the local server map, CSfM also estimates the unknown scale factor of the map from onboard visual SLAM system by comparing the difference between the relative poses of key frames computed onboard and in the server.

In CCM-SLAM[35] , each agent runs only visual odometry with limited number of key frames. The data of newly created key frames from the agents, including detected features, their descriptors, and 3D positions, are constantly sent to the server. The server constructs local maps from those key frames, trying to close the loop and merge the local maps by place recognition. Pose optimization and bundle adjustment are also applied in the server to further refine those maps. The agents constantly download the updated key frames from the server (even created from different agents) to augment the local map for better pose estimation.

The aforementioned systems rely on only vision sensors. While obtaining promising results in experimental tests, they may easily fail because of degradation of image quality when the environment does not contain enough textures or have a good illumination condition, when the cameras are rapidly moving. To improve the robustness, we may consider incorporating inertial sensors in the visual SLAM systems just like single-agent cases.

MOARSLAM[46] adopts a framework where the agent runs a monocular visual-inertial SLAM that leads to better accuracy and robustness. The loop closure running on the agent tries to match re-visited key frames stored in both the agent and the server. If the queried key frame is successfully matched in the server map, the system will download those surrounding key frames and integrate them into the local map on the agent. Similar to MOARSLAM[46], CVI-SLAM[36] also incorporates inertial measurements in the visual SLAM system. But each agent runs visual-inertial odometry only and send the key frame data to the server for mapping related tasks.

In [44], a collaborative SLAM system for several flying micro-aerial vehicles has been demonstrated as shown in (Figure 4). In this system, most of the computation is performed on a smartphone mounted on the drone, including state estimation, control and planning. The state estimation is achieved by running a

visual-inertial odometry on the phone. The key-frame data from individual phones are then sent to a centralized server to fuse those individual maps into a global map.

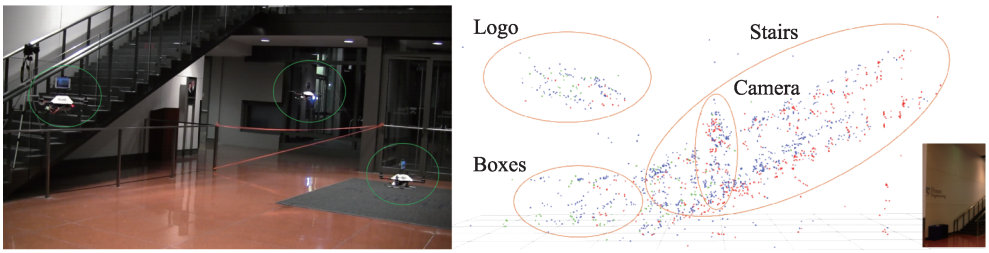

Figure 4 Three flying drones whose states are estimated collaboratively from smartphones[48]. The map fused from different phones is shown on the right.

4 Filter based approach vs Key-frame based approach

It is well known that there are two different frameworks for developing a visual SLAM system. One is based on Extended Kalman Filter (EKF), the other is based on key frame optimization. Both approaches have been applied in multi-agent visual SLAM systems as summarized in (Table 1). We discuss the two approaches to facilitate the discussion about the key components in the next section.

4.1 Filter based approach

In a filter-based visual SLAM system, the unknown variables are estimated by an extended Kalman filter. The state vector usually contains both the ego motion of the camera and the three-dimensional coordinates of the landmarks[51]. The camera pose estimation problem is solved through one iteration of the extended Kalman filter (including both the prediction and update steps). The iteration however also involves updating the 3D coordinates of those landmarks. It becomes inefficient when the number of landmarks is large. So the number of landmark features is limited to a small value in practice to ensure the real-time performance. This however usually leads to less accurate results than that of key-frame optimization approach[52].

One strategy to avoid high computation costs in a filter-based visual SLAM system is to remove the landmarks from the state and keep both the current camera pose and historical poses. The observation model is revised by eliminating the dependence on landmarks, and converted into a new observation model that relies only on pose variables. More specifically, instead of using the perspective projection model, this strategy employs feature tracks to derive temporal constrains on the camera poses at different time instances. This approach, known as multi-state constraint Kalman filter (MSCKF)[16], has become a major framework for designing a visual-inertial odometry system. Without the landmarks in the state vector, the update can be computed efficiently in spite of a great number of feature tracks involved. The comparison between the classic EKF approach and MSCKF approach can be better understood in (Figure 5).

4.2 Key-frame based approach

Key-frame based approach is the most popular framework for visual SLAM. As shown in the early work[10], it separates the camera pose tracking that requires real-time computation from the time-consuming mapping tasks in the key-frame based framework to make the visual SLAM system highly efficient. In this approach, camera pose estimation and mapping are solved alternatively. That means camera pose estimation is solved based on the result of mapping and vice versa. The key frame is a data structure that

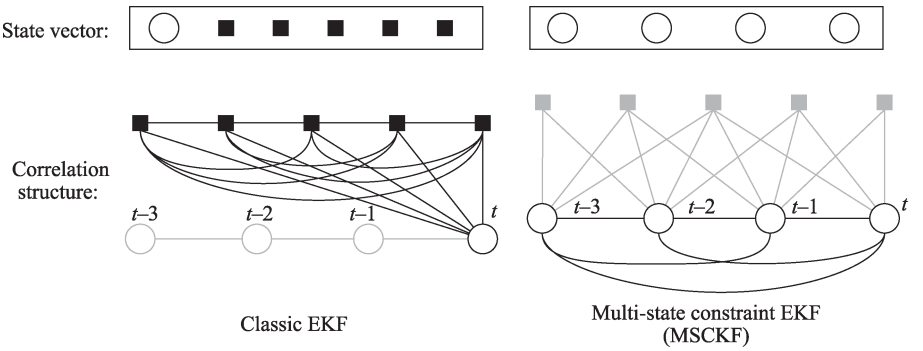

Figure 5 EKF approach[8] vs MSCKF approach[16]. The black squares are the landmarks and the white circles are the camera poses, while solid lines represent the correlation between different variables.

usually stores feature points detected on the image and their corresponding 3D points. With those 3D points from recent key frames and their corresponding feature points detected in the current image, we can solve the current camera pose by 3D-2D alignment, using Perspective-n-Point (PnP) algorithms[53,54] or directly solve this problem through nonlinear least squares optimization. Mapping is done separately between different key frames by triangulation of corresponding feature points. Bundle adjustment usually runs in a parallel thread to refine the key-frame poses and the 3D map from time to time. The difference between filter-based approach and key-frame based approach is illustrated in Figure 6.

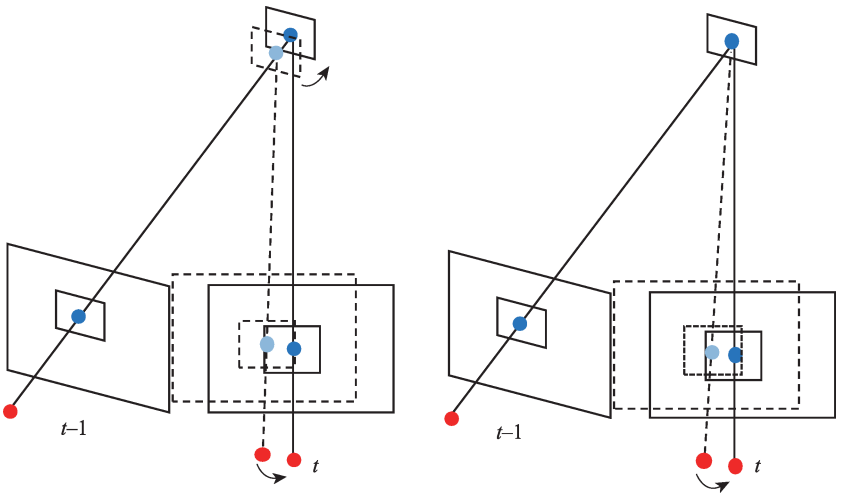

Figure 6 Pose estimation in filter based and key-frame approaches. Left: Filter based approach adjusts both the camera pose and 3D coordinates of the landmarks by EKF update; Right: key-frame based approach keeps the 3D coordinates as constants and estimate the current camera pose only. After that, key-frame based approach keeps the camera poses as constants and triangulates new 3D points from time to time.

Adopting whether the filter-based approach or a key-frame based approach largely depends on the specific applications. Generally speaking, filter-based approach is usually easier to be implemented and faster when the state dimension is well controlled or the MSCKF paradigm has been applied. Key-frame based approach usually leads to better accuracy because of its recursive optimization, but incurs significantly more computation than filter-based approach does. For application with limited computational resources, such as light-weight aerial vehicles and AR/VR headsets, filter-based approach may be more appropriate. But as the hardware is developing fast, key-frame based approaches could become more and more popular.

Key-frame based approaches are much easier to be extended to multi-agent settings. Most existing

methods adopting key-frame based approaches simply send the key frames from the individual agent to the back-end server to do mapping tasks collaboratively. By contrast, the filter-based approach is however not trivial to be applied to a multi-agent SLAM system to enable "collaboration" among different agents since it is not straightforward to deploy a single filter on different agents. Hence the systems that adopt the filter-based approach also extract key frames to facilitate map fusion or collaborative pose estimation[42,44].

5 Key components of a collaborative visual SLAM

Similar to those of monocular ones, the main components of a collaborative visual SLAM can also be roughly divided into tasks related to camera pose tracking and tasks related to mapping. In contrast to monocular cases, those tasks in a multi-agent visual SLAM usually involve processing data from different agents. For example, common features extracted from different cameras are used together to estimate the camera poses and generate the 3D points. This kind of collaboration usually leads to better accuracy and robustness. We named such operations involving information from different agents/cameras as inter-agent/camera operations. Those operations that involve information only from individual agent/camera are named intra-agent/camera operations. In what follows, we will discuss some ideas of implementing the key components in existing collaborative visual SLAM systems.

5.1 Collaborative pose estimation

Camera pose estimation is a fundamental task in a visual SLAM system. It requires to estimate 6DoF camera poses in real time. As aforementioned, camera pose estimation can be implemented by incorporating information from individual camera or different cameras. We named them as intra-camera pose estimation or inter-camera pose estimation.

Pose estimation is processed very differently in the two frameworks. Nevertheless, we will discuss the pose estimation method in both filter-based and key-frame based approaches in the following sections.

The key-frame data structure is flexible to sharing information among different agent. As in [35, 36] systems, the key frames are sent to the server and can be download by other agents. That means each agent is able to use key frames produced by other agents for pose estimation, leading to better accuracy and robustness. If the video cameras mounted on different agents are synchronized, an inter-camera pose estimation[14] using current video frames from different cameras can be applied. This pose estimation method takes both the static points and moving points into account and allows robust pose estimation even when there are a large portion of moving objects in the scene. The 3D motion of the feature points on the moving objects can be also obtained with multiple view geometry. This could be better understood in (Figure 7).

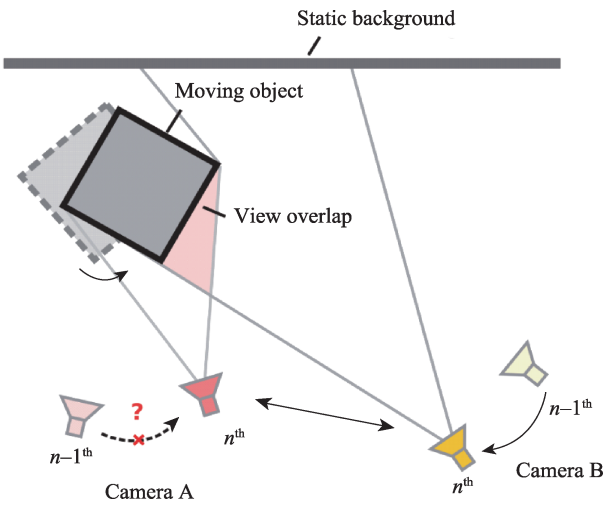

Figure 7 An illustration of inter-camera pose estimation[14]. The moving object occludes the static background from the viewpoint of Camera A. The camera B however is able to see both the moving object and the back ground.

Though the camera pose estimation is addressed quite differently in filter-based approaches and key-frame based approaches. Both approaches have been proved to be effective and efficient when applied in monocular visual SLAM systems.

But in the multi-agent setting, it is not straightforward for filter based approaches to exchange the map information among different agents, since the map stored in the state of the filter[8] is usually sparse and a large number of map points are discarded to restrict the dimension of the state. It hence requires an additional mapping component that works the same as the key-frame based approach to generate key-frame based maps.

5.2 Collaborative mapping

As aforementioned, probably the best way to represent a map in a multi-agent visual SLAM system is using a set of key frames, where each key frame stores 2D feature points with their descriptors and 3D coordinates, sometimes with the raw images. The camera poses of key frames and their topological connections are also stored, usually referred as *pose graph*[55]. Though there are other ways such as using stochastic map and 3D point clouds, the advantage of using a key-frame based map is that it compactly stores the necessary information for mapping related tasks such as map fusion and refinement.

Mapping tasks mainly involve triangulating new 3D points from corresponding feature points from different frames. However, in a multi-agent visual SLAM system, the mapping tasks can be further classified into *intra-camera mapping* and *inter-camera mapping*. The former is similar to the mapping tasks in a monocular visual SLAM system, which uses only video frames from only one camera. The latter tries to use corresponding points from images captured by different cameras to generate map points. As shown in[14], inter-camera will lead to better accuracy of relative pose estimation among different cameras. To perform inter-camera mapping, one should firstly find out if those cameras are looking at the same place, and then using feature matching to establish point correspondences among different cameras.

Another important mapping related task is loop closure. In fact, there are two kinds of loop closure: *intra-camera loop closure* and *inter-camera loop closure* in a multi-agent visual SLAM system. Intra-camera loop closure aims to find the revisited place for a single camera to reduce the drift error. Inter-camera loop closure attempts to detect the overlaps in some specific regions among multiple individual maps for map fusion.

Both types of loop closures follow the same pipeline. The first step is to recognize the revisited places. This process is usually referred as *place recognition*, which is done by evaluation of the similarity of two key frames. After two key frames have been successfully matched as the images captured from the same place, the second step is to compute the relative transforms between the two key frames and use the relative transforms as extra constraints to adjust the poses of those key frames. This is usually referred as *pose graph optimization*. After pose graph optimization, *bundle adjustment* is applied to further refine both the key-frame poses and the 3D coordinates of feature points. In the next section, we discuss about loop closure in multi-agent visual SLAM systems.

5.3 Loop closure

5.3.1 Place recognition & relative transform computation

Place recognition[56] is a fundamental task for detection of revisited area for loop closure or overlapping area for map fusion, through checking whether the two images are similar or not. A global descriptor is required to describe the image in order to match different images. For instance, we may use down-sampled images or tiny images as the global descriptors for comparison. For a given key frame, we may compare it with the tiny images of all key frames stored in the map[57]. If the cameras are moving in a small scene, this method is demonstrated to be highly efficient and effective. However, when the scene becomes large, it

requires to use some kinds of indexing approaches to ensure the real-time performance. Bag of words techniques[58] are commonly used in such tasks. A place recognition method used in a visual SLAM system is DBoW2[20]. This method uses the binary descriptors (BRIEF or ORB) and FAST corners as the feature points. Hence it is highly efficient. Comparing with Bag of words that adopt SIFT or SURF features, the DBoW2 matching method is about one order of magnitude faster. DBoW2 has become the dominant approach to place recognition in a single-agent visual SLAM system since the computational resource is often limited. However, as there is usually a powerful centralized server in a multi-agent system, we may consider to employ more advanced methods that are based on the deep neural networks[59].

We need to obtain the relative pose between two frames for the next step of optimization after two images are successfully matched. The relative pose computation can be classified into three categories. The first one is 2D-2D matching. In this approach, we consider using only the information of those 2D feature points on the images. Five-point algorithm[60] is applied to estimate the essential matrix and RANSAC is also adopt to handle outliers. The relative pose between two frames is then extracted from the essential matrix through matrix decomposition. However, the translation of the relative pose is only determined up to a scale. It therefore requires to treat the scale of translation vector as an unknown variable to be further estimated in the next optimization step.

The second approach is by 2D-3D matching[61], which exploits the fact that the key frames contain also corresponding 3D points. Like camera pose estimation, we can apply PnP algorithm to obtain the camera pose of one frame with respect to the 3D points from the other frame. In contrast to 2D-2D approach, the relative pose from 2D-3D approach is fully determined as a constraint for the next pose graph optimization.

The third approach is by 3D-3D matching, where the 3D point clouds are directly used for relative transform estimation. This approach is usually adopted in RGBD SLAM[62] and semi-dense visual SLAM systems[24]. The process of 3D-3D matching much resembles that of the 2D-2D matching, whereas the relative transform is computed directly from the two 3D point clouds after corresponding feature points have been founded. Another difference is that the relative transform is represented by a similarity transform instead of a rigid transform to account for the scale drift[61]. It has been demonstrated in [13] that by using similarity transform as relative transform constraints will lead to better address the scale drift problem.

5.3.2 Optimization

It requires to use optimization to refine both the camera poses and 3D structures after revisited places have been recognized for either intra-camera or inter-camera loop closure. Bundle adjustment is well known as the "Gold Standard" algorithm for the optimization problem of SfM. It is a nonlinear least squares solver applying Gaussian-Newton or Levenberg-Marquardt iterations, which may be easily trapped in a local minimum if the scale of the problem is large or the initial guess is not close to the real value. Hence we usually run pose graph optimization[63] before applying bundle adjustment so as to get a better initial guess.

The nodes of a pose graph represent the camera poses of those key frames. The edges are the relative transform constraints among different poses, usually represented by rigid transforms. The new edges introduced by place recognition are inserted to the pose graph. Due to the drift error or inconsistent coordinate frame, the newly inserted edges may conflict with old edges, leading to large inconsistent errors at the new edges. By taking all the constraints from both old and new edges together, pose graph optimization runs a nonlinear least squares solver to minimize the sum of squared constraint errors. Using rigid transform as pose constraints works well for correction of rotational and translational drifts, but not well for correction of scale drifts. Hence a better way is to use similarity transform as the pose

constraints[61]. It has been demonstrated in [13] that using similarity transforms could lead to better loop closure performance.

One important issue in pose graph optimization is that there may exist some outlier loop detections. The false loop detection may be caused by imperfect descriptors or perceptual aliasing. Perceptual aliasing means that different places may look very similar or exactly the same in scene. It happens frequently in daily environments. In such cases, false loop detections may exist even if the descriptor is perfectly designed. The outlier loop detections lead to incorrect relative pose constraints, hence will break the map if there are not well treated. To improve the robustness of pose graph optimization, we may increase the level of confidence for similarity matching by setting a high matching threshold. But this will significantly reduce the number of accepted loop detections. One way to find out the loop detection outliers is to exploit the prior about the relative transform between matched frames, which is available from the frontend on individual agent. The newly detected loops that are inconsistent with the predicted ones are treated as outliers. This method however will not work well for large loops because of the large uncertainty of prediction.

It becomes much more difficult to identify the outliers within inter-camera loop closures, since there is no prior information available about the relative pose between the matched frames. Undetected outliers will break the 3D structures after optimization, which could be hardly recovered. One possible solution is to collect a set of loop detections and apply pairwise consistency checking to them[64] for outlier identification. The pairwise consistency checking can be demonstrated in (Figure 8). If two loop measures are consistent, one of them can be predicted by the other one using the intra-camera relative transforms from the front end. Using such kind of checking, a maximum subset of consistent loop measurements can be found by the maximum clip algorithm[65] from graph theory.

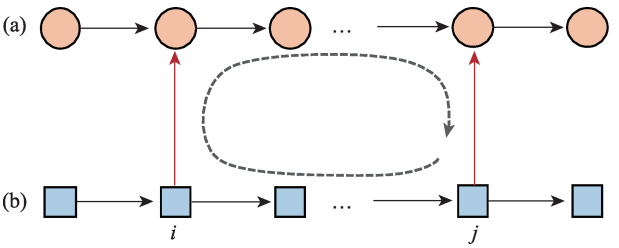

Figure 8 Consistency check of pairwise loop closure measurements between two robots. Red lines indicate the loop closure measurements; Black solid lines represent the relative transforms between successive frames.

Bundle adjustment is finally applied to further refine both the camera poses of key frames and 3D coordinates of point clouds after pose graph optimization. This process is the same as that in a single-agent visual SLAM system. Bundle adjustment[12,38] has been studied for many years. There are many open source implementations[66-68] for solving bundle adjustment problem efficiently. Some systems[36,44] use visual-inertial SLAM as the front end. In that case, bundle adjustment needs to incorporate also the inertial measurements in the cost function. The inertial measurements can be obtained by pre-integration[69] to avoid repeated integration that costs a lot of computation.

5.4 Feature detection & matching

Detecting and matching feature points across different images are essential for both camera tracking and mapping tasks. There are several choices for feature points. One of the most popular feature points used in early systems[37,42] are FAST corners[70]. Compared with other corner features like SIFT[71] or SURF[72], FAST corners are highly efficient to be extracted since it involves only a set of binary comparisons with adjacent pixels. Usually, those FAST corners are described by small image patches around the corner positions. With those image patches, the FAST corners can be tracked over frames as described in C2TAM[42] and MOARSLAM[46]. An alternative way to describe a FAST corner is using ORB features[73] instead of directly

using the image patches. The ORB descriptor is an extension of BRIEF descriptor[74] with rotation invariance and robust to large view changes. It has been demonstrated in [13] that ORB feature is a reliable feature for pose estimation, mapping and loop detection.

If we want to get more stable feature extraction and tracking results, we may consider using the Kanade-Lucas-Tomasi (KLT) features[75] and use optical flow[76] to track those features. Though it involves more computational effort, however it can run efficiently with GPU acceleration as shown in [14, 39]. Using optical flow for feature tracking only works properly when the two images are close. In other words, it is only suitable for feature tracking in successive video frames captured by the same camera. To match the corresponding feature points across different viewpoints, usually a descriptor that is robust to large viewpoint change requires to be used. Though SIFT descriptor[71] is probably the most powerful feature descriptor, more efficient descriptors such as SURF[72] and ORB[73] are used for efficiency.

Another way to improve the efficiency of feature matching is through guided search. Such idea has been employed since the early visual SLAM systems. In a filter-based visual SLAM system[8,16,77], the guided search is performed by firstly computing the range of possible locations of a landmark predicted from the current state variables. The range is usually decided by the prediction uncertainty, or the projection covariance matrix.It therefore reduces the searching area from the whole image to a small area surrounding the predicted location. The guided search in a key-frame based system is performed in a similar way but with an additional guided search using the updated camera pose for generating new map points. As the 3D coordinates of the unmapped point is unknown, the additional guided search is conducted along the epipolar line by comparing the image patches between the given point and the candidate point. In PTAM[10] and ORB-SLAM[13], epipolar search is performed to triangulate new points among different key frames. In CoSLAM[14], epipolar search is applied to different cameras to generate new map points.

Recently direct approaches[24,25] have gained much attention in visual SLAM community because of its advantages on both accuracy and efficiency. Though it may be a good choice to run a direct approach on each agent, it is difficult to directly match pixels among different cameras to establish correspondences. Hence if direct approaches are applied to multi-agent visual SLAM, more attention should be payed to the feature matching problem across different cameras.

5.5 Outlier rejection

Outlier rejection is critical to camera pose estimation. There two approaches to addressing the outlier rejection problem. The first one is through RANdom SAmple Consensus (RANSAC) algorithm[78,79]. By randomly sampling a minimum set of 2D-3D corresponding points, we apply Perspective-n-Point (PnP) algorithm to compute the camera pose and check the consistency of the remaining corresponding points using the current pose estimate. The pose estimate from the minimum set that leads to the best consistency among all those random samples is used as final result. Those inconsistent corresponding points are removed as outliers. For pose estimation, the minimum number of corresponding 2D-3D points is three. However, a larger number is usually used to achieve better estimates in practice.

Though RANSAC is robust to a large portion of outliers, or to sometimes severe outliers, it usually leads non-negligible computation costs especially the number of elements in the sampling set is large. Sometimes we may use other sensors like inertial measurement units (IMU)[41] or state prediction[77,80] to reduce the size of the sampling set. The second approach to outlier rejection is applying a robust estimator[81] for camera pose estimation. Comparing with RANSAC, robust estimator usually consumes less computation costs. The robust estimator can be easily implemented by a slight revision to an existing

nonlinear least squares optimizer. The robust estimator has been widely used in key-frame based visual systems. It however may easily fail when severe outliers or a large number of outliers present, as well as sometimes the initial estimate is not close to the real one. Hence sometimes we may combine RANSAC and robust estimator to make the system more robust. For example, we may run the robust estimator in the first pass and check the re-projection error after pose estimation. Usually, if the estimator fails, the re-projection error becomes noticeable large. In that case, RANSAC may be run in the second pass to rescue the pose estimation from failure.

It would be interesting to consider the outliers produced by the moving objects in the scene. There are in fact two kinds of outliers. The first kind of outliers are mainly caused by false feature matching, which in turn produces incorrect 2D-3D correspondences for pose estimation. The second kind of outliers come from the moving points in dynamic environments that contain moving objects. In this case, the 2D-3D correspondences may be correct, but since the 3D points are moving, their 3D coordinates are in fact changing overtime. If we still treat them as static points, it will lead to inconsistent pose estimates. However, if we have multiple cameras with different viewpoints toward the target points, we can simultaneously update the 3D coordinates of those moving points and estimate the camera poses that are consistent with the static backgrounds as shown in Figure 7.

But how can we identify the dynamic points from those real outliers? By exploring how points satisfy the geometric constraint in both temporal and spatial domains, the dynamic points and real outliers can be identified. This is based on the observation that, a static point should always meet the geometric constraint in both temporal and spatial domains given that the camera poses are correctly estimated. One rule to check if the geometric constraint is violated is to examine the re-projection error on the image. For a static point, if its 2D-3D correspondences are correctly obtained, the re-projection errors at all video frames, no matter from the same camera or different cameras, should be small values. As the moving points satisfy only the geometric constraint across different cameras at the same time instance, while violating the geometric constraint among different time instances, we may check the re-projection errors accordingly to identify the moving points. A pipeline for identify static points, moving points, and real outliers has been presented in [14].

5.6 System initialization

There are two different ways for map initialization. The first way is done by each agent independently. A typical approach to map initialization is to use five-point algorithm[72] to the relative pose between the first two key frames and triangulate the initial 3D points from the corresponding feature points. We may also combine five-point algorithm with homography estimation to handle degenerated motion as described in [13] . Each agent initializes a local map independently. The local maps are fused once the overlaps between them are found by place recognition. The disadvantage of this approach is that there would be no global map for task planning for a long period before map overlaps being detected. The second way is done by all agents together as shown in [14]. This approach requires all the agents look at the same place in the scene. Their relative poses are estimated from the images captured by different agents. The process is much like solving a structure-from-motion problem. We may use either a sequential SfM solver as in [14] or a global SfM solver [82,83] to get both the initial poses of all cameras and the initial 3D point cloud.

6 Decentralized architecture

Most of existing multi-agent visual SLAM system adopt a centralized architecture with a powerful server

processing time-consuming computational tasks related to map optimization such as loop detection, pose graph optimization, and bundle adjustment. Sometimes such centralized server may not be available or the communication is only available among neighbor agents within limited distance. In such cases, it is better to deploy a decentralized architecture.

It is much more difficult to deploy a decentralized architecture than a centralized one to a multi-agent visual SLAM system. The major challenge is that the time-consuming computational tasks requires be distributed to different agents with limited onboard computational resources. It therefore involves designing complicate distributed algorithms to solve those computational tasks appropriately, which is not easy particularly for the map optimization tasks such as pose graph optimization, bundle adjustment. It is also important to consider the communication load to design the distributed algorithms. Different implementations could lead to totally different amount of information exchanged among agents as demonstrated in (Figure 9). We need to consider the strict bandwidth constraints when applying decentralized architecture. Another challenge is about the data consistency and synchronization among different agents. This issue becomes increasingly complicate with the growing number of agents. All aforementioned challenges prevent the decentralized architecture from being widely used in multi-agent visual SLAM systems, nevertheless there are still some notable progresses made in recent years.

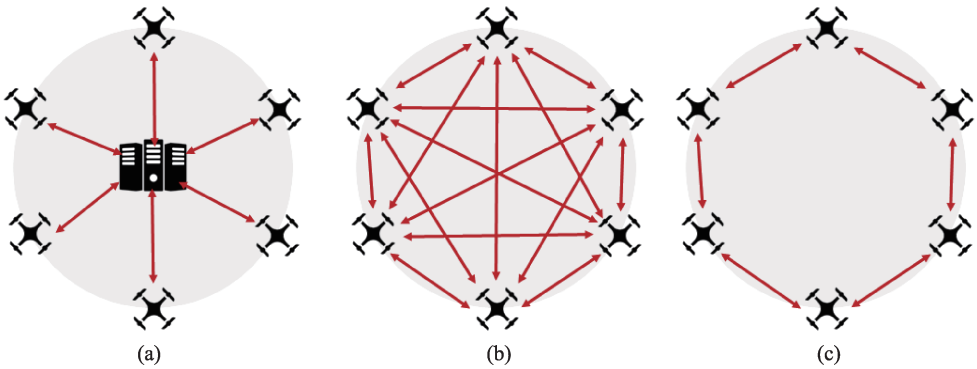

Figure 9 (a) Centralized architecture used in most existing multi-agent visual SLAM systems, whose communication load is linear to the number of agents; (b) Naïve implementation of a decentralized architecture, where each agent needs to communicate with all other agents; (c) A more efficient architecture, where each agent only communicates with nearby agents.

The decentralized architecture present in DDF-SAM[84] allows the robots to exchange their summary map to each other. The summary map, comprising the commonly observed landmarks, is obtained by marginalization of non-shared variables of the local map. Each robot receives summary maps from neighboring robots (including the summary map from itself) and constructs a neighborhood map to solve the relative transforms between different local maps. Each local map is however optimized independently without using the marginalization information from others to avoid double-counting measurements. The architecture in DDF-SAM2.0[85] combines the neighborhood map and the local map together and adopt a down-dating scheme to address the double-counting issue.

Though the architectures in DDF-SAM or DDF-SAM2.0 have been tested in planar scenarios with 2D landmarks measured by a range-bearing sensor, they have shortcomings to be applied to vison cases. The first one is that the number of variables to be marginalized could be extremely large. For example, there may be tens of thousands feature points in the local map, but only hundreds of them can be commonly observed by other robots. It therefore incurs a great amount of computational load to make the system

highly inefficient. The second shortcoming is that the neighborhood map is construct and optimized at every agent. Construction of the neighborhood map means that the marginal information of neighboring robots needs to be transmitted to each other, leading to a high communication cost. Optimization at every agent is also a waste of computation resources, which becomes even worse if the number of agents involved becomes large.

Until recently the decentralized architecture was applied to visual SLAM systems. In [86], a novel distributed algorithm is proposed to solve the pose-graph optimization problem. The idea is to adopt the iterative solver such as Successive Over-Relaxation (SOR) and Jacobi Over-Relaxation (JOR)[87] to solve the normal equation within iterations of the nonlinear least-squares optimization. Unlike direct solvers by matrix decomposition, iterative solvers are naturally suitable for distributed processing. It is demonstrated in [86] that the state of each robot can updated independently with only the Hessian sub-matrices related to neighboring robots. This approach has been applied to visual SLAM as shown in [88]. However, it still remains an open problem to develop distributed algorithms to another important task of map optimization.

7 Conclusion

In this article, we focus on multi-agent SLAM problem using the video cameras as the primary sensory inputs. After a brief history of visual SLAM techniques was described, a taxonomy of existing methods of multi-agent visual SLAM was presented, which includes the systems developed in the past ten years. As we can see, most existing approaches adopt a centralized architecture, where usually a powerful server is in charge of time-consuming mapping tasks by collecting key frame data from each agent. The map optimization results are sometimes sent back to each agent to improve the performance of the front-end visual SLAM. We find out that either pose estimation and mapping tasks in a multi-agent visual SLAM system can be divided into intra-camera operations or inter-camera operations, where the inter-camera operations are what we call "collaboration". Based on this observation, we discuss the ideas of implementing several key components: collaborative pose estimation, collaborative mapping, loop closure, feature extraction and matching, outlier rejection, and system initialization. We then discuss about the decentralized architecture where the communication load should be carefully addressed. The most challenging issue in decentralized architecture is how to design efficient distributed algorithms to solve the inter-camera operation or map optimization problems. Though we have witnessed a notable progress has been made in some aspects in recent years, further research is still required to enable decentralized architecture being applied to real-world cases.

Currently there might be two major areas to which multi-agent visual SLAM could be applied. The first area is the robotics. An excellent survey on multi-agent SLAM on robotics systems has been presented[89]. The recent survey paper[90] offers a throughout review on key techniques on swarm aerial robots, including modeling, control, and sensing. As we can see in [90], multi-agent visual SLAM has shown a great potential in both mapping and state estimation in GNSS-denied environments. The other area is VR and AR. Involving multiple users in a VR/AR application has been receiving increasingly interest in recently years[91], such that multi-agent visual SLAM will be an important topic in the next stage of SLAM research.

In conclusion, this survey aims to provide researchers an overall view of multi-agent visual SLAM. There are still many related topics or research problems not covered or not deeply investigated in this survey. Nevertheless, we hope our survey could provide a good starting point for the future research on multi-agent visual SLAM.

References

1 Fuentes-Pacheco J, Ruiz-Ascencio J, Rendón-Mancha J M. Visual simultaneous localization and mapping: a survey. Artificial Intelligence Review, 2015, 43(1): 55–81
DOI:10.1007/s10462-012-9365-8

2 Scaramuzza D, Fraundorfer F. Tutorial: visual odometry. IEEE Robotics and Automation Magazine, 2011, 18(4), 80–92

3 Klinker K, Berkemeier L, Zobel B, Wüller H, Huck-Fries V, Wiesche M, HartmutRemmers, OliverThomas, Krcmar H. Structure for innovations: A use case taxonomy for smart glasses in service processes. Multikonferenz Wirtschaftsinformatik Lüneburg, Deutschland. 2018

4 Siegwart R, Nourbakhsh I R, Scaramuzza D. Introduction to autonomous mobile robots. MIT press, 2011

5 Whyte H D. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. Robotics and Automation Magazine, 2006, 1–9

6 Bailey T, Durrant-Whyte H. Simultaneous localization and mapping (SLAM): part II. IEEE Robotics & Automation Magazine, 2006, 13(3): 108–117
DOI:10.1109/mra.2006.1678144

7 Schonberger J L, Frahm J M. Structure-from-motion revisited. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA, IEEE, 2016, 4104–4113
DOI:10.1109/cvpr.2016.445

8 Davison A J, Reid I D, Molton N D, Stasse O. MonoSLAM: real-time single camera SLAM. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29(6): 1052–1067
DOI:10.1109/tpami.2007.1049

9 Welch G, Bishop G. An introduction to the Kalman Filter: Department of Computer science. University of North Carolina, 2006

10 Klein G, Murray D. Parallel tracking and mapping for small AR workspaces. In: 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality. Nara, Japan, IEEE, 2007, 1–10
DOI:10.1109/ismar.2007.4538852

11 Schonberger J L, Frahm J M. Structure-from-motion revisited. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, 4104–4113

12 Triggs B, McLauchlan P F, Hartley R I, Fitzgibbon A W. Bundle adjustment—A modern synthesis//Vision Algorithms: Theory and Practice. Berlin, Heidelberg: Springer, 2000, 298–372
DOI:10.1007/3-540-44480-7_21

13 Mur-Artal R, Montiel J M M, Tardos J D. ORB-SLAM: A versatile and accurate monocular SLAM system. IEEE Transactions on Robotics, 2015, 31(5): 1147–1163
DOI:10.1109/tro.2015.2463671

14 Zou D P, Tan P. CoSLAM: collaborative visual SLAM in dynamic environments. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2013, 35(2): 354–366
DOI:10.1109/tpami.2012.104

15 Liu H M, Zhang G F, Bao H J. Robust keyframe-based monocular SLAM for augmented reality. In: 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct). Merida, Yucatan, Mexico, IEEE, 2016
DOI:10.1109/ismar-adjunct.2016.0111

16 Mourikis A I, Roumeliotis S I. A multi-state constraint kalman filter for vision-aided inertial navigation. In: Proceedings 2007 IEEE International Conference on Robotics and Automation. Rome, Italy, IEEE, 2007, 3565–3572
DOI:10.1109/robot.2007.364024

17 Li M Y, Mourikis A I. High-precision, consistent EKF-based visual-inertial odometry. The International Journal of Robotics Research, 2013, 32(6): 690–711
DOI:10.1177/0278364913481251

18 Triggs B, McLauchlan P F, Hartley R I, Fitzgibbon A W. Bundle adjustment—A modern synthesis//Vision Algorithms: Theory and Practice. Berlin, Heidelberg: Springer, 2000, 298–372

DOI:10.1007/3-540-44480-7_21

19 Kaess M, Johannsson H, Roberts R, Ila V, Leonard J J, Dellaert F. ISAM2: Incremental smoothing and mapping using the Bayes tree. The International Journal of Robotics Research, 2012, 31(2): 216–235
DOI:10.1177/0278364911430419

20 Galvez-López D, Tardos J D. Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics, 2012, 28(5): 1188–1197
DOI:10.1109/tro.2012.2197158

21 Straub J, Hilsenbeck S, Schroth G, Huitl R, Möller A, Steinbach E. Fast relocalization for visual odometry using binary features. In: 2013 IEEE International Conference on Image Processing. Melbourne, VIC, Australia, 2013, IEEE, 2548–2552

22 Williams B, Klein G, Reid I. Real-time SLAM relocalisation. In: IEEE 11th International Conference on. Rio de Janeiro, Brazil, IEEE, 2007, 1–8

23 Newcombe R A, Lovegrove S J, Davison A J. DTAM: Dense tracking and mapping in real-time. In: 2011 International Conference on Computer Vision. Barcelona, Spain, IEEE, 2011
DOI:10.1109/iccv.2011.6126513

24 Engel J, Schops T, Cremers D. LSD-SLAM: Direct Monocular SLAM. Computer Vision–ECCV, 2014

25 Engel J, Koltun V, Cremers D. Direct Sparse Odometry. IEEE transactions on pattern analysis and machine intelligence, 2017, 40(3), 611–625

26 Forster C, Pizzoli M, Scaramuzza D. SVO: Fast semi-direct monocular visual odometry. In: 2014 IEEE International Conference on Robotics and Automation. Hong Kong, China, IEEE, 2014
DOI:10.1109/icra.2014.6906584

27 Forster C, Zhang Z C, Gassner M, Werlberger M, Scaramuzza D. SVO: semidirect visual odometry for monocular and multicamera systems. IEEE Transactions on Robotics, 2017, 33(2): 249–265
DOI:10.1109/tro.2016.2623335

28 Leutenegger S, Lynen S, Bosse M, Siegwart R, Furgale P. Keyframe-based visual–inertial odometry using nonlinear optimization. The International Journal of Robotics Research, 2015, 34(3): 314–334
DOI:10.1177/0278364914554813

29 Qin T, Li P L, Shen S J. VINS-mono: A robust and versatile monocular visual-inertial state estimator. IEEE Transactions on Robotics, 2018, 34(4): 1004–1020
DOI:10.1109/tro.2018.2853729

30 Zou D P, Wu Y X, Pei L, Ling H B, Yu W X. StructVIO: visual-inertial odometry with structural regularity of man-made environments. IEEE Transactions on Robotics, 2019, 35(4): 999–1013
DOI:10.1109/tro.2019.2915140

31 Liu H M, Chen M Y, Zhang G F, Bao H J, Bao Y Z. ICE-BA: incremental, consistent and efficient bundle adjustment for visual-inertial SLAM. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, UT, USA, IEEE, 2018, 1974–1982
DOI:10.1109/cvpr.2018.00211

32 Shafi M, Molisch A F, Smith P J, Haustein T, Zhu P Y, de Silva P, Tufvesson F, Benjebbour A, Wunder G. 5G: A tutorial overview of standards, trials, challenges, deployment, and practice. IEEE Journal on Selected Areas in Communications, 2017, 35(6): 1201–1221
DOI:10.1109/jsac.2017.2692307

33 Howard A. Multi-robot simultaneous localization and mapping using particle filters. The International Journal of Robotics Research, 2006, 25(12): 1243–1256
DOI:10.1177/0278364906072250

34 Kim B, Kaess M, Fletcher L, Leonard J, Bachrach A, Roy N, Teller S. Multiple relative pose graphs for robust cooperative mapping. In: 2010 IEEE International Conference on Robotics and Automation. Anchorage, AK, USA, IEEE, 2010
DOI:10.1109/robot.2010.5509154

35 Schmuck P, Chli M. CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization

and mapping for robotic teams. Journal of Field Robotics, 2019, 36(4): 763−781
DOI:10.1002/rob.21854

36 Karrer M, Schmuck P, Chli M. CVI-SLAM: collaborative visual-inertial SLAM. IEEE Robotics and Automation Letters, 2018, 3(4): 2762−2769
DOI:10.1109/lra.2018.2837226

37 Castle R, Klein G, Murray D W. Video-rate localization in multiple maps for wearable augmented reality. In: 2008 12th IEEE International Symposium on Wearable Computers. Pittaburgh, PA, USA, IEEE, 2008
DOI:10.1109/iswc.2008.4911577

38 Agarwal S, Snavely N, Seitz S, Szeliski R. Bundle adjustment in the large. In: Computer Vision—ECCV. Berlin, Heidelberg, Springer, 2010

39 Perron J M, HuangR, ThomasJ, Zhang L, Tan P, Vaughan R T. Orbiting a moving target with multi-robot collaborative visual slam. In: Workshop on Multi-View Geometry in Robotics. 2015, 1339−1344

40 Forster C, Lynen S, Kneip L, Scaramuzza D. Collaborative monocular SLAM with multiple Micro Aerial Vehicles. In: IEEE International Conference on Intelligent Robots and Systems. Tokyo, Japan, IEEE, 2013, 3963−3970

41 Kneip L, Scaramuzza D, Siegwart R. Robust Real-Time Visual Odometry with a Single Camera and an IMU. In: Proceedings of the British Machine Vision Conference, 2011

42 Riazuelo L, Civera J, Montiel J M M. C2TAM: A Cloud framework for cooperative tracking and mapping. Robotics and Autonomous Systems, 2014, 62(4): 401−413
DOI:10.1016/j.robot.2013.11.007

43 Civera J, Davison A J, Montiel J M M. Interacting multiple model monocular SLAM. In: 2008 IEEE International Conference on Robotics and Automation. Pasadena, CA, USA, IEEE, 2008, 3704−3709
DOI:10.1109/robot.2008.4543779

44 Loianno G, Mulgaonkar Y, Brunner C, Ahuja D, Ramanandan A, Chari M, Diaz S, Kumar V. A swarm of flying smartphones. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems. Daejeon, South Korea, IEEE, 2016, 1681−1688
DOI: 10.1109/IROS.2016.7759270

45 Loianno G, Mulgaonkar Y, Brunner C, Ahuja D, Ramanandan A, Chari M, Diaz S, Kumar V. Smartphones power flying robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Hamburg, Germany, IEEE, 2015, 1256−1263
DOI: 10.1109/IROS.2015.7353530

46 Morrison J G, Gálvez-López D, Sibley G. MOARSLAM: multiple operator augmented RSLAM//Springer Tracts in Advanced Robotics. Tokyo, Japan, Springer, 2016, 119−132
DOI:10.1007/978-4-431-55879-8_9

47 Keivan N, Patron-Perez A, Sibley G. Asynchronous adaptive conditioning for visual-inertial SLAM//Experimental Robotics. Cham: Springer International Publishing, 2015, 309−321
DOI:10.1007/978-3-319-23778-7_21

48 Burri M, Nikolic J, Gohl P, Schneider T, Rehder J, Omari S, Achtelik M W, Siegwart R. The EuRoC micro aerial vehicle datasets. The International Journal of Robotics Research, 2016, 35(10): 1157−1163
DOI:10.1177/0278364915620033

49 Castle R, Klein G, Murray D W. Video-rate localization in multiple maps for wearable augmented reality. In: 2008 12th IEEE International Symposium on Wearable Computers. Pittaburgh, PA, USA, IEEE, 2008, 15−22
DOI:10.1109/iswc.2008.4911577

50 Castle R, Klein G, Murray D W. Video-rate localization in multiple maps for wearable augmented reality. In: 2008 12th IEEE International Symposium on Wearable Computers. Pittaburgh, PA, USA, IEEE, 2008
DOI:10.1109/iswc.2008.4911577

51 Davison A J, Reid I D, Molton N D, Stasse O. MonoSLAM: real-time single camera SLAM. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29(6): 1052−1067
DOI:10.1109/tpami.2007.1049

52 Strasdat H, Montiel J M M, Davison A J. Real-time monocular SLAM: Why filter? In: 2010 IEEE International

Conference on Robotics and Automation. Anchorage, AK, USA, IEEE, 2010
DOI: 10.1109/ROBOT.2010.5509636

53 Quan L, Lan Z D. Linear N-point camera pose determination. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999, 21(8): 774–780
DOI:10.1109/34.784291

54 Lepetit V, Moreno-Noguer F, Fua P. EPnP: an accurate O(n) solution to the PnP problem. International Journal of Computer Vision, 2009, 81(2): 155–166
DOI:10.1007/s11263-008-0152-6

55 Grisetti G, Kummerle R, Stachniss C, Burgard W. A Tutorial on Graph-Based SLAM. IEEE Intelligent Transportation Systems Magazine, 2010, 2(4): 31–43
DOI:10.1109/MITS.2010.939925

56 Lowry S, Sünderhauf N, Newman P, Leonard J J, Cox D, Corke P, Milford M J. Visual Place Recognition: A Survey. IEEE Transactions on Robotics, 2016, 32(1): 1–19
DOI:10.1109/TRO.2015.2496823

57 Klein G, Murray D. Improving the agility of keyframe-based SLAM//Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, 802–815
DOI:10.1007/978-3-540-88688-4_59

58 Nister D, Stewenius H. Scalable recognition with a vocabulary tree. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. New York, NY, USA, IEEE, 2015, 1063–6919
DOI:10.1109/cvpr.2006.264

59 Arandjelovic R, Gronat P, Torii A, Pajdla T, Sivic J. NetVLAD: CNN architecture for weakly supervised place recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, NV, USA, IEEE, 2016, 5297–5307
DOI:10.1109/cvpr.2016.572

60 Nistér D. An efficient solution to the five-point relative pose problem. IEEE transactions on pattern analysis and machine intelligence, 2004, 26(6), 756–777

61 Strasdat H, Davison A J, Montiel J M M, Konolige K. Double window optimisation for constant time visual SLAM. In: 2011 International Conference on Computer Vision. Barcelona, Spain, IEEE, 2011, 2352–2359
DOI:10.1109/iccv.2011.6126517

62 Endres F, Hess J, Engelhard N, Sturm J, Cremers D, Burgard W. An evaluation of the RGB-D SLAM system. In: 2012 IEEE International Conference on Robotics and Automation. St Paul, MN, USA, IEEE, 2012, 1691–1696
DOI:10.1109/icra.2012.6225199

63 Kummerle R, Grisetti G, Strasdat H, Konolige K, Burgard W. G2o: A general framework for graph optimization. In: 2011 IEEE International Conference on Robotics and Automation. Shanghai, China, IEEE, 2011, 3607–3613
DOI:10.1109/icra.2011.5979949

64 Mangelson J G, Dominic D, Eustice R M, Vasudevan R. Pairwise consistent measurement set maximization for robust multi-robot map merging. In: 2018 IEEE International Conference on Robotics and Automation. Brisbane, QLD, IEEE, 2018, 2916–2923
DOI:10.1109/icra.2018.8460217

65 Pattabiraman B, Patwary M M A, Gebremedhin A H, Liao W K, Choudhary A. Fast algorithms for the maximum clique problem on massive graphs with applications to overlapping community detection. Internet Mathematics, 2015, 11(4/5): 421–448
DOI:10.1080/15427951.2014.986778

66 Lourakis M I, Argyros A A. SBA: A software package for generic sparse bundle adjustment. ACM Transactions on Mathematical Software, 2009, 36(1), 2
DOI:10.1145/1486525.1486527

67 Wu C, Agarwal S, Curless B, Seitz S M. Multicore bundle adjustment. In: CVPR. Providence, RI, USA, 2011, 3057–3064
DOI: 10.1109/CVPR.2011.5995552

68 Agarwal S. Mierle K. Others, Ceres Solver. 2012

69 Forster C, Carlone L, Dellaert F, Scaramuzza D. On-manifold preintegration for real-time visual: inertial odometry. IEEE Transactions on Robotics, 2017, 33(1): 1–21
DOI:10.1109/tro.2016.2597321

70 Rosten E, Drummond T. Machine learning for high-speed corner detection. In: European conference on computer vision. Berlin, Heidelberg, Springer, 2006

71 Lowe D G. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 2004, 60(2): 91–110
DOI:10.1023/b:visi.0000029664.99615.94

72 Bay H, Ess A, Tuytelaars T, van Gool L. Speeded-up robust features (SURF). Computer Vision and Image Understanding, 2008, 110(3): 346–359
DOI:10.1016/j.cviu.2007.09.014

73 Rublee E, Rabaud V, Konolige K, Bradski G. ORB: An efficient alternative to SIFT or SURF. In: 2011 International Conference on Computer Vision. Barcelona, Spain, IEEE, 2011
DOI:10.1109/iccv.2011.6126544

74 Calonder M, Lepetit V, Strecha C, Fua P. BRIEF: Binary robust independent elementary features. In: Computer Vision—ECCV 2010. Berlin, Heidelberg, Springer, 2010, 778–792
DOI:10.1007/978-3-642-15561-1_56

75 Shi J B. Good features to track. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94. Seattle, WA, USA, IEEE, 1994, 1063–6919
DOI:10.1109/cvpr.1994.323794

76 Lucas B D, Kanade T. An iterative image registration technique with an application to stereo vision, 1981, 674–679

77 Zhou H Z, Zou D P, Pei L, Ying R D, Liu P L, Yu W X. StructSLAM: visual SLAM with building structure lines. IEEE Transactions on Vehicular Technology, 2015, 64(4): 1364–1375
DOI:10.1109/tvt.2015.2388780

78 Hartley R, Zisserman A. Multiple view geometry in computer vision. Cambridge: Cambridge University Press, 2004
DOI:10.1017/cbo9780511811685

79 Nistér D. Preemptive RANSAC for live structure and motion estimation. Machine Vision and Applications, 2005, 16(5): 321–329
DOI:10.1007/s00138-005-0006-y

80 Civera J, Grasa O G, Davison A J, Montiel J M M. 1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry. Journal of Field Robotics, 2010, 27(5): 609–631
DOI:10.1002/rob.20345

81 Wilcox R. Robust regression//Introduction to Robust Estimation and Hypothesis Testing. Elsevier, 2012, 471–532
DOI:10.1016/b978-0-12-386983-8.00010-x

82 Jiang N J, Cui Z P, Tan P. A global linear method for camera pose registration. In: 2013 IEEE International Conference on Computer Vision. Sydney, Australia, IEEE, 2013, 481–488
DOI:10.1109/iccv.2013.66

83 Cui Z P, Tan P. Global structure-from-motion by similarity averaging. In: 2015 IEEE International Conference on Computer Vision. Santiago, Chile, IEEE, 2015, 864–872
DOI:10.1109/iccv.2015.105

84 Cunningham A, Paluri M, Dellaert F. DDF-SAM: Fully distributed SLAM using constrained factor graphs. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. Taipei, Taiwan, IEEE, 2010, 3025–3030
DOI: 10.1109/IROS.2010.5652875

85 Cunningham A, Indelman V, Dellaert F. DDF-SAM 2.0: Consistent distributed smoothing and mapping. In: 2013 IEEE International Conference on Robotics and Automation. Karlsruhe, Germany, IEEE, 2013, 5220–5227
DOI:10.1109/icra.2013.6631323

86 Choudhary S, Carlone L, Nieto C, Rogers J, Christensen H I, Dellaert F. Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models. The International Journal of Robotics

Research, 2017, 36(12): 1286–1311
DOI:10.1177/0278364917732640

87 Saad Y. Iterative methods for sparse linear systems. Society for Industrial and Applied Mathematics, 2003
DOI:10.1137/1.9780898718003

88 Cieslewski T, Choudhary S, Scaramuzza D. Data-efficient decentralized visual SLAM. In: 2018 IEEE International Conference on Robotics and Automation. Brisbane, QLD, Australia, IEEE, 2018, 2466–2473
DOI:10.1109/icra.2018.8461155

89 Saeedi S, Trentini M, Seto M, Li H. Multiple-robot simultaneous localization and mapping: A review. Journal of Field Robotics, 2016, 33(1): 3–46
DOI:10.1002/rob.21620

90 Chung S J, Paranjape A A, Dames P, Shen S J, Kumar V. A survey on aerial swarm robotics. IEEE Transactions on Robotics, 2018, 34(4): 837–855
DOI:10.1109/tro.2018.2857475

91 Egodagamage R, Tuceryan M. Distributed monocular visual SLAM as a basis for a collaborative augmented reality framework. Computers & Graphics, 2018, 71: 113–123
DOI:10.1016/j.cag.2018.01.002