Hindawi
Journal of Sensors
Volume 2017, Article ID 6842173, 11 pages
https://doi.org/10.1155/2017/6842173

Research Article

Distributed Monocular SLAM for Indoor Map Building

Ruwan Egodagamage and Mihran Tuceryan

Indiana University-Purdue University Indianapolis, Indianapolis, IN, USA

Correspondence should be addressed to Ruwan Egodagamage; rjegodag@iupui.edu

Received 28 April 2017; Revised 26 June 2017; Accepted 3 July 2017; Published 10 August 2017

Academic Editor: Jacky C. K. Chow

Utilization and generation of indoor maps are critical elements in accurate indoor tracking. Simultaneous Localization and Mapping (SLAM) is one of the main techniques for such map generation. In SLAM an agent generates a map of an unknown environment while estimating its location in it. Ubiquitous cameras lead to monocular visual SLAM, where a camera is the only sensing device for the SLAM process. In modern applications, multiple mobile agents may be involved in the generation of such maps, thus requiring a distributed computational framework. Each agent can generate its own local map, which can then be combined into a map covering a larger area. By doing so, they can cover a given environment faster than a single agent. Furthermore, they can interact with each other in the same environment, making this framework more practical, especially for collaborative applications such as augmented reality. One of the main challenges of distributed SLAM is identifying overlapping maps, especially when relative starting positions of agents are unknown. In this paper, we are proposing a system having multiple monocular agents, with unknown relative starting positions, which generates a semidense global map of the environment.

1. Introduction

Utilization of indoor maps is a critical component of accurate indoor tracking when existing infrastructures such as GPS do not work reliably. Therefore, generating such maps with high accuracy for unknown environments becomes critical in the infrastructure of indoor tracking. Furthermore, such maps may be generated partially by different agents moving in and out of an environment, and unifying these independently and partially generated maps into a highly accurate global map is crucial. The map generation and its utilization for localization can be done in many different modalities. Simultaneous Localization and Mapping (SLAM) is one of the main techniques for such map generation. In modern applications, multiple mobile agents may be involved in the generation of such maps, thus requiring a distributed computational framework.

SLAM is a problem that addresses generating a map of an environment and tracking an agent in the environment. These two tasks are interrelated, since an accurate map is necessary to localize the agent precisely, and only a correctly localized agent can construct a good map. The SLAM problem is also known as the Tracking and Mapping (TAM) problem.

Cameras are becoming a popular choice for SLAM, since they are ubiquitous in smart devices. Furthermore, the smaller form factor and the lower cost of cameras also contribute to this choice. When we use a camera as the input device, the process is called visual SLAM. For visual SLAM, three main types of cameras are used: monocular, stereo, and RGBD. Unlike monocular cameras, stereo and RGBD cameras provide depth data in addition to image data to simplify the initialization and the pose estimation process.

Visual SLAM uses either the direct or the feature-based methods. Direct methods work on the intensity information of images without computing features and generally produce denser maps. Dense maps can be more attractive in certain applications, such as augmented reality, in which a user is interacting with the environment and virtual objects in the environment. It is desirable that this interaction be realistic and seamless. A dense map of the environment makes this interaction possible.

In distributed SLAM, multiple agents perform SLAM in an environment collaboratively. These agents (which are cameras for the purposes of this paper) can enter and exit the environment at any time. If there is a map of the environment, the agents can utilize it to localize themselves in it. If an agent moves in a part of the environment that is not mapped, it can start building the map and localize itself in it as part of the SLAM process. Each agent can do this independently, however, when they are operating in a common environment, it makes sense to use their locally built maps to complement each other. At the same time, they can complete and improve the global map, while helping each other in their respective tasks.

Additionally, using multiple agents to perform SLAM increases the robustness of SLAM process, which makes it more fault tolerant and less vulnerable to catastrophic failures. One of the main challenges in distributed SLAM is to compute map overlaps, especially when agents have no prior knowledge of their relative starting positions. Usually, agents also have limited bandwidth to communicate with each other.

In this paper, we introduce a distributed framework for monocular visual SLAM agents with no prior knowledge of their relative positions.

2. Related Work

In a seminal paper Smith et al. [1] introduced a solution to the SLAM problem using extended Kalman filter (EKF-SLAM). In their work, the extended Kalman filter is used to estimate the posterior distribution over agent pose and landmark positions incrementally. However, processing a covariance matrix is a significant challenge as it grows with the number of landmarks. The entire covariance matrix has to be updated even when the system observes one landmark. This severely limits the number of landmarks in EKF-SLAM, typically a few hundred. Furthermore, EKF-SLAM has Gaussian noise assumptions. FastSLAM by Montemerlo et al. [2, 3] addressed above limitations using a Monte Carlo Sampling (particle filter) based approach. Most importantly FastSLAM supported nonlinear process models and non-Gaussian pose distributions. In more recent work, FastSLAM by Cain and Leonessa [4] uses a compressed occupancy grid to reduce the data usage of each particle by 40%. Pei et al. [5] used distributed unscented particle filter to avoid reconfiguring the entire system during vehicle state estimation. Martinez-Cantin and Castellanos in [6] proposed an Unscented Kalman Filter based approach (UKF-SLAM) to support large scale environments.

Davison et al. [7] introduced MonoSLAM, a SLAM method of capturing the path of a freely moving camera (6 Degrees of Freedom) while generating a sparse map. This monocular visual SLAM method worked in a room-sized environment. The map consisted of image patches representing features. Their solution was a combination of EKF-SLAM for estimation and Particle Filtering (PF) for feature initialization. The entire system is initialized by positioning the camera in front of a marker.

Klein and Murray in [8] presented Parallel Tracking and Mapping (PTAM), one of the most significant solutions for visual SLAM. This robust SLAM solution mainly focused on accurate and fast mapping in a similar environment to MonoSLAM. Its implementation decoupled mapping and localization into two threads. The front-end thread only performs pose estimation and feature tracking while the back-end thread performed mapping and everything else, such as feature initialization and removing unnecessary key frames. A set of sparse point features represented the map. The system is initialized by moving the camera roughly 10 centimeters perpendicular to the optical path. RANSAC [9] and 5-point algorithm [10] initialized the system. A global bundle adjustment (BA) [11] with Levenberg-Marquardt optimization [10] adjusted the pose of all key frames. Furthermore, a local BA changed the pose of a subset of key frames to allow a reasonable rate of exploration.

Although MonoSLAM and PTAM address the same problem, PTAM used BA in contrast to MonoSLAM's incremental approach. BA is heavily used and proven to work well for offline Structure from Motion (SfM). Even though BA is relatively computationally expensive, PTAM and other researchers recently adopted BA for many real-time monocular visual SLAM solutions. Strasdat et al.'s analysis in [12] showed that increasing the number of image features acquired per frame is more beneficial than incorporating information from increased number of closely placed camera frames. They argue that the former increases the accuracy of the motion estimation and a better map estimation for a given computational budget. Their analysis hence favors bundle adjustment techniques over incremental methods for accurate monocular visual SLAM. Moreover, BA helps to increase the number of features on the map, leading to denser maps.

Scale drift is one of the biggest challenges in monocular visual SLAM. Strasdat et al. [13] introduced a pose graph optimization technique that corrects the scale drift at loop closures. Their method handled large looped trajectories well.

The work by DTAM by Newcombe et al. [14] and LSD-SLAM by Engel et al. [15, 16] utilize image pixel intensities directly instead of computed features for SLAM. Their systems generate dense or semidense maps of the environment. Furthermore, these direct methods are more robust to motion blur of images.

During the SLAM process, an agent might revisit the same location in multiple instances. Error accumulation can lead this to go unnoticed. The solution for this problem is referred to as loop closing. This could be done using appearance based image-to-image, map-to-map, or map-to-image matching approaches. The survey from Williams et al. [17] concludes with positive remarks on map-to-image approaches.

2.1. Distributed SLAM. One of the challenges in generating a globally consistent map is identifying map overlaps of agents. It is relatively easier to determine map overlaps if all of the agent relative poses are known at all times. For example, Nettleton et al. [18] used global positioning sensors (GPS) to detect agent locations. When the agent position is known, it is only a matter of doing a proximity check between agent trajectories to detect their map overlaps. However, location

sensors like GPS are not always available, and they do not do well in indoors, nor in underwater vehicles.

The relative transformations between coordinate systems of agent maps can be computed if the starting position of each agent is known. Paull et al. [19] initialized all agents from known locations. Next, agents performed SLAM and estimated their new locations, while at the same time communicating their locations to each other. Given that the agents already knew the transformation between their maps, they were able to easily determine map overlaps, similar to the case in having location sensors.

When these agent relative locations are unknown, the distributed SLAM problem becomes more challenging. In some contributions, agents continued to build local maps until they saw each other. Howard et al. [20] proposes a method in which each agent would be able to detect other agents. Agents use these coincidental encounters to find their relative locations. Dieter et al. in [21] presents a method where each agent is actively seeking other agents in the environment to find relative locations between them. These methods either require special sensors to be seen by each other or to actively seek each other.

Some methods heavily depend on a central node. Zou and Tan in [22] allowed cameras to move independently in a dynamic environment. However, all of their cameras were initialized from the same scene and connected to the same computer. Although their cameras were distributed, all frames were processed at the same time in their SLAM process. This tightly couples agents, since agents do not possess any knowledge of the environment individually.

The multiagent system by Forster et al. in [23] used a centralized ground station for mapping, loop closure detection, and map merging. However, relying on a central agent is highly prone to failures, especially when the central node fails.

In [24], agents used a master-slave approach where the slave is always in the master agent's map to maintain a map overlap. Their method restricts the free movement of the slave agent.

Williams et al. in [25, 26] introduced a method to construct a global map from a multiagent system using a Constrained Local Submap Filter (CLSF). In their method, overlaps between the local and global maps are determined using a Maximal Common Subgraph (MCS) method. Kin and Newman in [27] use a visual similarity matrix to determine relative agent locations. A subsequence of visually similar images is detected from the images captured in each agent. Cunningham et al. in [28, 29] formulate the distributed SLAM problem using a graphical model. In their fully decentralized system, each agent maintained a consistent local map, augmented with information shared in a neighborhood of agents. In [30], authors proposed a fully distributed solution for the data association problem.

In the system proposed in [30], local feature matches are propagated through the low-bandwidth communication network. This method helps agents to find global correspondences with other agents with no direct connections. Work done in [18] discusses most informative features to transmit, to reduce bandwidth requirements.

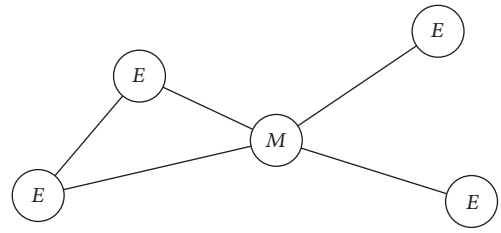


Figure 1: The network of nodes: the exploring nodes (E) are connected to a monitoring node (M). Some exploring nodes are connected with each other.

Our proposed framework performs distributed SLAM with no knowledge of the initial agent locations. Furthermore, the agents do not get their location directly from sensors like GPS. Instead, they estimate the location using a visual SLAM process. Moreover, the framework does not rely on a central agent, but rather a network of monitoring agents that look for map overlaps of SLAM agents.

We used the experimental framework for distributed SLAM we introduced in [31], during the development of this framework.

3. Materials and Methods

3.1. Nodes of the Distributed Framework. Our distributed SLAM framework consists of two types of nodes, the *exploring node* and the *monitoring node*. Each node is deployed in its own physical machine. At any given time, the framework has at least one monitoring node and an arbitrary number exploring nodes. Each node is identified using a global unique identifier.

A connection between nodes is made when nodes are required to communicate with each other over the network. As shown in Figure 1, the exploring nodes are connected to a monitoring node. Furthermore, two exploring nodes become connected when their maps overlap with each other.

3.1.1. Exploring Node. Each exploring node performs a semi-dense visual SLAM by using a camera as the only sensor, based on the work by [16]. Our choice is based on the fact that denser maps describe the environment in more detail, compared to the sparse, feature-based maps. Denser maps enable better interaction with the environment, especially in AR applications. This also means exploring nodes that communicate more data, compared to feature-based methods. Each exploring node maintains a set of key frames and a pose graph to represent the map. It periodically sends out its map information to the monitoring node, as well as to the other exploring nodes to which it is connected. Furthermore, it processes incoming commands from the monitoring node.

3.1.2. Monitoring Node. The monitoring node's responsibilities include map overlap detection between the exploring nodes and loop closure detection in each exploring node. It maintains a feature store in which all *salient* features are stored. Features from all incoming key frames are matched against the feature store to identify map overlaps and loop

```
(1) procedure getUniqueID (key frame_id, node_id)
(2)     id ← ShiftLeft(node_id, 20)
(3)     id ← id + keyframe_id
(4)     return id                          ▷ A globally unique identifier
(5) end procedure
```

Algorithm 1: Unique identifier for a key frame.

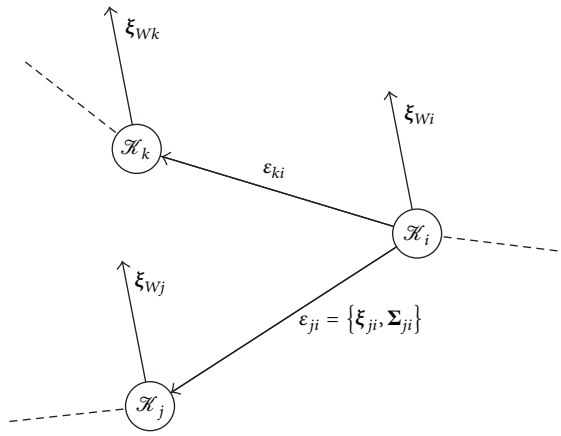


Figure 2: Pose graph with key frames and similarity transform constraints.

closures. The monitoring node uses a graph, which will be referred to as the *fusion graph* in this paper, to prioritize and issue commands to merge overlapping maps.

In advanced configurations of our proposed distributed framework, there could be multiple monitoring nodes. Ideally, each monitoring node connects to overlapping exploring nodes. In practice, monitoring nodes may move exploring nodes among themselves dynamically to minimize the number of overlapping exploring node clusters. Map overlap detection between two exploring nodes belonging to different monitoring nodes is accomplished by sharing features between monitoring nodes. In this paper, our experiments are limited to a single monitoring node configuration.

3.2. Map of the Environment by Exploring Node. As shown in Figure 2, the exploring node maintains a map of the environment using multiple key frames and a pose graph.

3.2.1. Key Frames. The ith *key frame*, $\mathcal{K}_i$, consists of an *absolute pose* $\xi_{Wi} \in \mathbb{R}^7$, an image I_i, a map containing z coordinate reciprocals corresponding to nonnegligible intensity gradient pixels D_i (an inverse depth map), an inverse depth variance map V_i, and a list of features F_i. The absolute pose is encoded with a translation, along with orientation and scale parameters using a quaternion. The elements of ξ_{ji} are the three components of the translation and the four components of the quaternion representing the rotation. The scale is represented using the magnitude of the quaternion. When $\mathcal{K}_i$ is first introduced into the pose graph, the features of $\mathcal{K}_i$ are computed. In $\mathcal{K}_i$, i corresponds to a 32-bit globally unique identifier computed using Algorithm 1. Figure 3 contains a visual representation of two different key frames.

We used SURF [32] features and SIFT [33] descriptors in our framework. Because we computed features only for the key frames, the added computational cost that resulted did not adversely affect the real-time performance.

3.2.2. Pose Graph. Pose graph edges ε_{ji} contain similarity transformations ξ_{ji} and Σ_{ji} constraints. Here $\xi_{ji} \in \mathbb{R}^7$ and Σ_{ji} are relative pose transformations and corresponding covariance matrix between ith and jth the key frames, respectively.

Both absolute pose ξ_{Wi} and similarity transformation ξ_{ji} are encoded with a translation (three components) and orientation with scale using a quaternion (four components).

3.2.3. Generating and Updating the Map. The SLAM process simultaneously tracks the camera against the current key frame $\mathcal{K}_i$ and improves its D_i and V_i based on its new observations. Once the camera deviates significantly from the $\mathcal{K}_i$, either a new key frame is created or, if available, an existing key frame is selected from the map. Next, if a new key frame is created, the previous key frame used for tracking is inserted into the pose graph. The pose graph is continuously optimized in the background. More information on the LSD-SLAM process is found in [15].

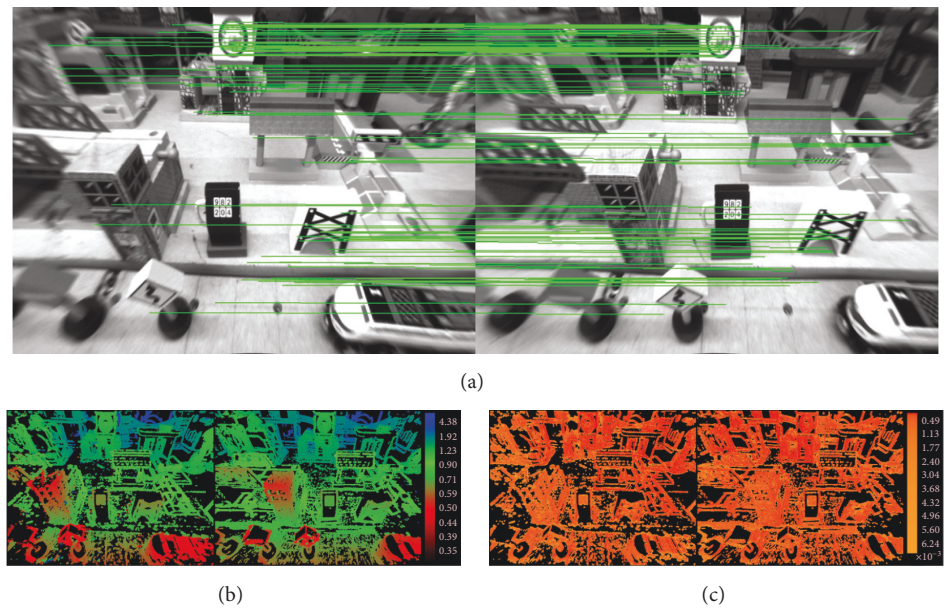
3.2.4. Identifying Salient Features. To determine the saliency of a feature, first the feature is filtered for its $V_i(X_p)$, where X_p is the location feature found. The pth feature in $\mathcal{K}_i$ should satisfy

$$V_i\left(X_p\right) < T \times D_i\left(X_p\right)^2, \tag{1}$$

where T is a threshold computed empirically. Only salient features are kept as F_i. We experimented with different values for T to minimize the number of features exchanged, while still achieving sufficient map overlap detection. We found 0.001 to be a good value with satisfactory results.

3.2.5. Sending Salient Features to Monitoring Node. For every salient feature in F_i, the corresponding 3D location X_p and the descriptor d_p are computed. Next, the key frame identifier i, the salient features $(X_p, d_p)_i$, and the pose ξ_{Wi} are sent to the monitoring node.

The communication process between the nodes is explained in more detail in Section 3.6.2.



(a)

(b) (c)

FIGURE 3: We show the matched features between key frames $\mathcal{K}_i$ and $\mathcal{K}_j$ superimposed on the images I_i and I_j (a). We also show the pseudo-color encoded D_i and D_j (b) and pseudo-color encoded V_i and V_j (c).

3.3. Map Overlap Detection in Monitoring Node. Exploring nodes of our distributed framework do not know their relative pose at the beginning. The monitoring node is responsible for detecting map overlaps and computing corresponding relative pose between nodes.

3.3.1. Feature Store. Each entry in the feature store contains a feature descriptor d_p, key frame identifier i, 3D feature location X_p, and key frame pose $\boldsymbol{\xi}_{Wi}$. Every incoming feature descriptor is matched against the entries in the feature store, to identify common features between two key frames. We used the FLANN [34] method for feature matching. We used a search radius of 150, which was calculated empirically. We divide each key frame into a grid of 16 equal sized cells. If features are matched in at least 10 cells with another key frame $\mathcal{K}_j$, it is concluded that there is an overlap between key frames $\mathcal{K}_i$ and $\mathcal{K}_j$. Our approach expects the indoor environment to contain sufficient textured regions of overlap to function properly. It fails if the overlapping region contains only scenes like texture-less walls.

If overlapping key frames belong to the same exploring node, it is considered that a loop closure is found. Otherwise, matching information contributes to the *fusion graph*.

3.3.2. Fusion Graph. All available exploring nodes are represented as vertices in the fusion graph as shown in Figure 4. Assume there is an overlap between key frames $\mathcal{K}_r$ and $\mathcal{K}_s$ and $\mathcal{K}_r \in e_i^{\mathcal{K}}$ and $\mathcal{K}_s \in e_j^{\mathcal{K}}$, where $e_i^{\mathcal{K}}$ represent key frames in ith exploring node. Then, the fusion graph contains an edge between e_i and e_j. The number of features matched between e_i and e_j is represented using c_{ij} as shown in Figure 4. Note that the edge between e_i and e_j could represent matching features between many different key frame pairs. If the direction of the edge is $e_j \rightarrow e_k$, map merging process is initialized from e_j. As shown in Figure 5, first e_j sends the map to e_k. Then, e_k merges the map and sends its original map to e_j. e_j merges the received map and notifies the monitoring node about the completion of map merging process. Assume that the fusion graph edge having the largest c_{ij} satisfies

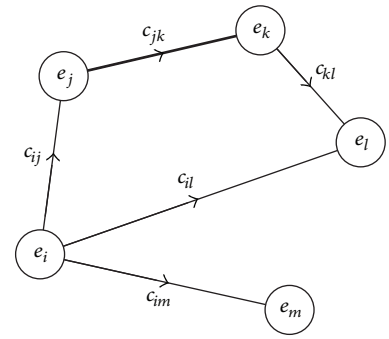


FIGURE 4: The fusion graph showing exploring nodes (e_i) and the number of matching features (c_{ij}) as the weight of each edge. In this example, c_{jk} is higher than other edges (indicated by the thicker edge), so e_j and e_k are merged first. Furthermore, the map merging process is initialized by sending e_j's map to e_k following the direction of the edge.

$$\max\left(c_{ij}\right) > m, \tag{2}$$

where m is an empirical threshold. Then the monitoring node concludes that a map overlap exists between exploring nodes e_i and e_j. Empirically, 120 shared features are found to be a good value for m. Next, the rigid transformation between e_i

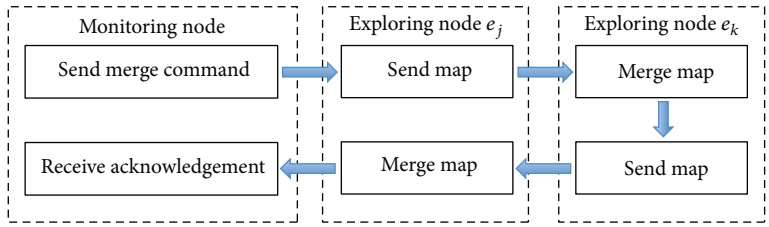


Figure 5: The map merging process of the fusion graph edge $e_j \rightarrow e_k$.

and e_j, $\boldsymbol{\xi}_{ji}$, is computed using a Singular Value Decomposition (SVD) based on the least squares method [35]. Similar to the absolute pose representation $\boldsymbol{\xi}_{ji}$ is encoded using 3 components of translation and 4 components of quaternion. The scale is initially assumed to be 1 and a proper value is estimated later, during the map merging followed by pose graph optimization in each exploring node. X_p of all relevant features between e_i and e_j are used for the computation. The RANSAC algorithm [9] is used to make the computation robust to outliers. Figure 3 shows a set of matched features between two key frames, $\mathcal{K}_i$ and $\mathcal{K}_j$.

3.3.3. Issuing Commands to Exploring Nodes. A *map merge* command is issued to exploring nodes e_i and e_j. The command contains the relative pose $\boldsymbol{\psi}_{ji}$ between two nodes. The command also contains the key frame correspondences used to compute the relative pose between e_i and e_j. Similarly, a *loop closure* command is issued to an exploring node e_s, when both overlapping key frames $\mathcal{K}_i$ and $\mathcal{K}_j$ belong to e_s. The command contains the relative pose $\boldsymbol{\xi}_{ji}$ between key frames, which is also computed using the same least squares method [35].

The communication process is explained in more detail in Section 3.6.2.

3.4. Merging Maps of Two Exploring Nodes. First, as shown in Figure 1, a connection is created between two exploring nodes. Once the connection is made, each exploring node sends its map to its counterpart. Once the map is received, the key frame correspondences found in the *map merge* are directly transformed into new constraints between pose graphs of e_i and e_j. The similarity transformation of the constraint is computed using key frame pose $\boldsymbol{\xi}_{Wk}$ and relative pose $\boldsymbol{\xi}_{ji}$ between exploring nodes.

Figure 6 shows how e_i and e_j were generating their own maps before merging. Figure 7 shows the resulting merged map for e_i. Once map merging is complete, each exploring node listens to its counterpart for new key frames and the pose graph, to incrementally update its map.

3.5. Loop Closure. In most instances, completing smaller loop closures increases the robustness of tracking. Completing large loop closures, however, has more impact in generating an accurate map. Direct semidense SLAM operations alone do not support large loop closures.

Upon receiving a loop closure command with $\boldsymbol{\xi}_{ji}$, the exploring node checks whether $\mathcal{K}_i$ and $\mathcal{K}_j$ are consecutive key frames in the pose graph. If that is the case, we discard the loop closure command since $\mathcal{K}_j$ was constructed using $\mathcal{K}_i$ and already has a better estimate for the edge ε_{ji}. Otherwise,

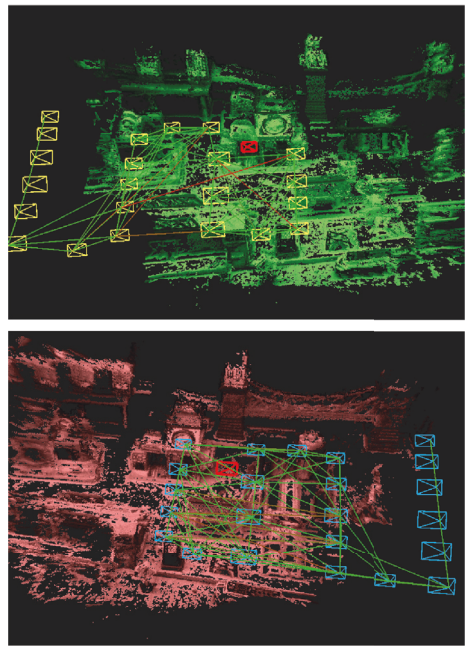


Figure 6: Map generation process of two exploring nodes. Each exploring node has its own coordinate system. Relative transformations between coordinate systems are initially not known.

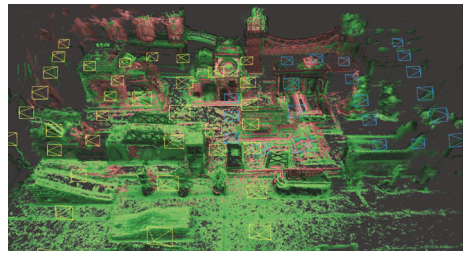


Figure 7: Resultant map of an exploration node after the map merging process. The exploring node's map and key frames are shown in green and yellow, respectively. The map and key frames received from the other node are shown in pink and blue, respectively. Constraints of the pose graph are not shown here to avoid too much clutter in the figure.

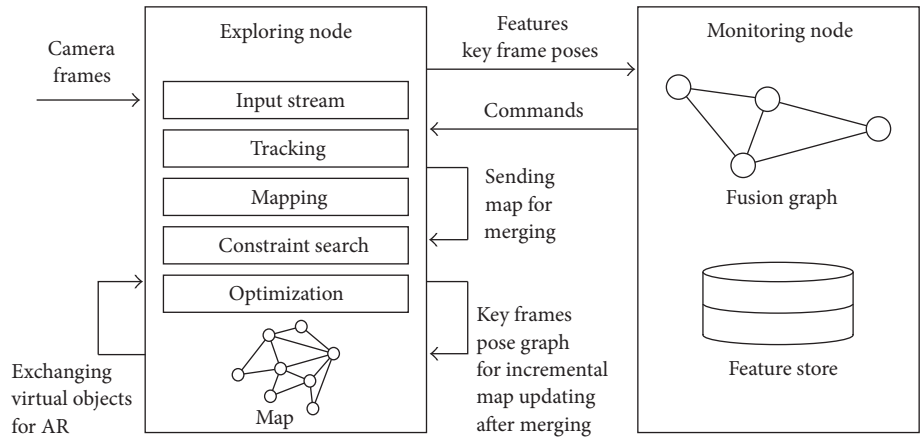


Figure 8: The distributed framework. In the figure, the arrows looping back to the exploring node rectangle represent communication between two exploring nodes.

it inserts the new edge and completes the loop closure process by performing another iteration of pose graph optimization.

3.6. System Implementation. We use the Robot Operating System (ROS) infrastructure to implement the distributed SLAM framework described in this paper [36]. A ROS node is responsible for performing computations. ROS also provides a message passing communication framework between nodes. Nodes in our framework are implemented as ROS nodes.

In its communication framework, ROS provides named communication buses called *topics*. Multiple nodes can publish messages to a topic while multiple subscribed nodes could receive them. ROS nodes can communicate with each other peer-to-peer via topics. In our framework, communication channels are implemented using ROS topics.

3.6.1. Components of the Distributed Framework. Figure 8 shows components of the distributed framework and the communications between nodes. The exploring node consists of five main modules: input stream, tracking, mapping, constraint search, and optimization modules. Each of these modules runs in its own thread and it contains the map.

The *input stream* module accepts all incoming messages including image frames, key frames, pose graph, map, and commands. All image frames are transferred to the tracking module. Key frames, pose graph, and map are transferred to the optimization module so that they can be merged into the map before an optimization iteration. Commands are processed in the input stream module itself.

The *tracking* module accepts the new frame from input stream module and tracks it against the current key frame. If the current key frame can no longer be used to track the current frame, a new key frame is created. The old key frame will be added to the map by the *mapping* module. The *constraint search* module is used to recover from tracking failures. The *optimization* module continuously optimizes the pose graph in the background.

The monitoring node maintains the feature store and the fusion graph as explained in Section 3.3.

3.6.2. Communication between Nodes. The distributed framework provides multiple communication channels for nodes. These communication channels are shown as arrows in Figure 8.

Upon creating new key frames, exploring nodes send salient features and the absolute pose of the key frame through the *features* channel. The monitoring node receives them and processes them to issue commands through the *commands* channel. The command could be either a *merge* command or a *loop closure* command.

When an exploring node receive a *merge* command, it creates multiple channels with the other exploring node. The *map* channel is used to exchange a map between each other. This channel ceases to exist once the map is transferred. And *key frames* and *pose graph* channels are created between these nodes.

For every new key frame, its information is written into the *key frames* channel. After every pose graph optimization, the pose graph information is written into *pose graph* channel. All exploring nodes that are using these channels incrementally update their maps after merging.

4. System Evaluation and Discussion

4.1. Public Datasets. To evaluate our framework, we need a monocular visual SLAM dataset, with multiple trajectories covering a single scene. We considered publicly available datasets, but they did not satisfy our requirements. For example, the dataset EuRoC [37] contains pure rotations, which did not work well with the monocular SLAM approach we used. The dataset Kitti [38] is mainly a stereo dataset. Even when we considered a single camera, the direct monocular SLAM process failed since the camera motion is along the optical axis. The TUM-Mono [39] dataset does not provide the ground truth for all frames and is primarily suitable for

Table 1: DIST-Mono dataset.

Dataset	Path	Initial camera rotation	Total travel (mm)
S01-A-0	Path A	0°	3706
S01-A-P20	Path A	20° CW	3706
S01-B-0	Path B	0°	3706
S01-B-N20	Path B	20° CCW	3706
S01-C-0	Path C	0°	3080

Figure 9: Experimental setup showing a camera mounted on a CNC machine allowing us to capture ground truth information. Camera mounted on a CNC machine.

evaluating single agent SLAM. That said, we created the DIST-Mono dataset to evaluate our framework. We also made it publicly available (http://slam.cs.iupui.edu).

4.2. Experimental Setup. Our experimental setup is designed to precisely define the ground truth of a camera motion. As shown in Figure 9 we mounted a Point Grey Firefly MV global shutter camera onto a Computer Numeric Controller (CNC) machine. We also prepared a 1 m × 1.5 m scene containing wooden objects. We then moved the camera along a path for about four minutes each time, while capturing its location ground truth periodically.

Our in-house built, 3-axis CNC machine is controlled using an open-source controller called TinyG. The controller converts the provided trajectory from the gcode file format into linear synchronized movements along x, y, and z axes. The maximum travel volume of the machine is 1 m × 1 m × 0.3 m ($x \times y \times z$). We also prepared a 1 m × 1.5 m scene containing wooden objects. The scene is uniformly lit by two 4-feet long LED tube lights.

To collect datasets, we moved the camera at a speed of 25 mm/s, along a path for about four minutes each time, while capturing its location ground truth periodically. We captured 640 × 480 resolution camera frames at 60 Hz and ground truth at 40 Hz. The CNC machine has 0.2 mm accuracy in all three axes. We developed an open-source ROS node (http://github.com/japzi/rostinyg) to capture the ground truth from the TinyG CNC controller.

During experimentation we played back the datasets at twice the speed it was recorded (2x), effectively making the camera movement to be 50 mm/s.

4.3. The DIST-Mono Dataset. The dataset consists of five subdatasets. We defined three camera motion paths: Path A, Path B, and Path C. All of these paths are on a plane slanted above the scene as shown in Figure 10(a). These paths have roughly 10% overlap with each other and three different starting points per path. We generated two datasets using Path A by rotating the camera around its z-axis. In S01-A-0, the camera optical axis and scene Y axis are on a vertical plane. In S01-A-P20, we rotated the camera around its y-axis by 20°. This is illustrated in Figure 10(b). Similarly, we created datasets S01-B-0, S01-B-N20, and S01-C-0 as shown in Table 1.

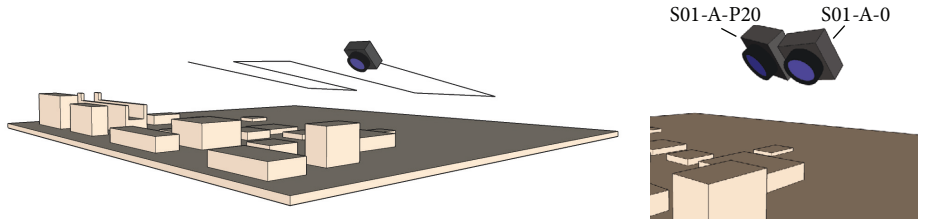
Table 2: Experiments and their absolute translation RMSE against ground truth.

Experiment	Datasets	RMSE (m)
Experiment 1	S01-A-0, S01-B-0	0.0136
Experiment 2	S01-A-0, S01-B-N20	0.0192
Experiment 3	S01-B-0, S01-C-0	0.0097
Experiment 4	S01-A-0, S01-C-0	0.0121

4.4. System Evaluation. As an experiment, we deployed two exploring nodes and one monitoring node in three different machines. One exploring node processed the dataset SCENE-A-0 and the other the dataset SCENE-B-N20. After map merging, each exploring node exported its key frame poses in TUM dataset [40] pose format. Most importantly, these poses contain key frames from both exploring nodes. We then computed the absolute translation RMSE [40] against the ground truth. To support the nondeterministic nature of the distributed framework, we ran the experiment five times and the median result is recorded. Similarly we performed two more experiments with other combinations of datasets as shown in Table 2. Given the fact that monocular visual SLAM systems do not capture the scale, we manually calculated that in all experiments to minimize the RMSE error.

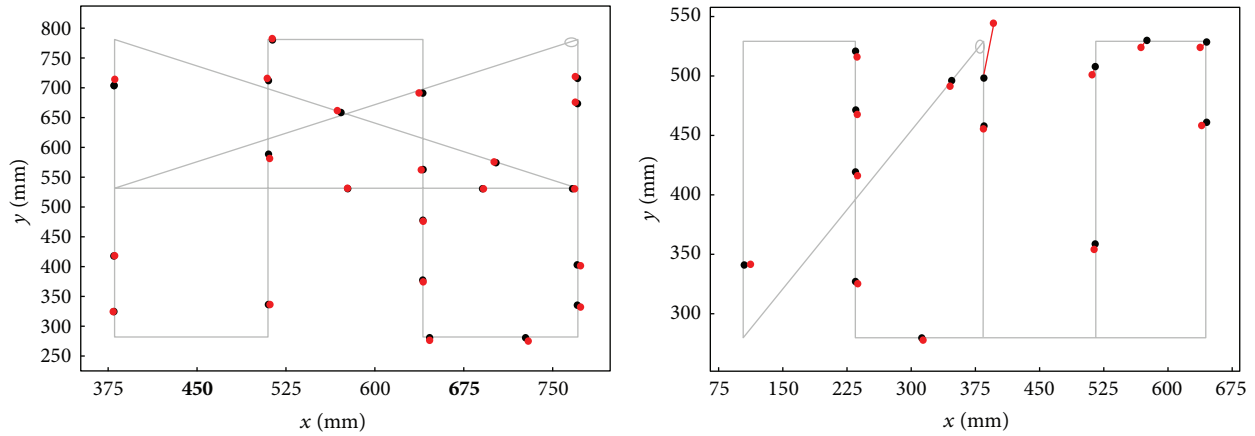
Figure 11 shows how estimated key frame positions are compared against ground truth in experiment 3. The red circles in the figure display the estimated key frame position, whereas the black circles display the ground truth of said key frame position. Red lines show the difference between the estimated and the ground truth positions of the key frame.

4.5. AR Application: Adding and Viewing Virtual Objects. We developed an AR application to test our framework. We added an AR window to each exploring node. The AR window allows users to add virtual objects (simple cube in our example) into its map. This allows us to demonstrate



(a) Motion paths are in a plane slanted above the scene (b) 20° clockwise rotation

Figure 10: Camera motion and its initial rotation for datasets.



(a) First exploring node (b) Second exploring node

Figure 11: Key frame position estimation against ground truth.

the collaborative AR potential of the distributed SLAM framework, in which (i) each agent is able to view the augmented scene from its viewpoint and (ii) if it is in an unexplored part of the scene, generate its own local map and contribute it to the global map.

We also added a relevant channel to share the virtual object information between exploring nodes. Given that the relative transformation between nodes is known for connected exploring nodes, these cubes are placed correctly in the map. Figure 12 shows AR windows of two exploring nodes and two interactively added cubes.

4.6. Discussion. Our framework relies on image features for map overlap detection. As explained in Section 3.2.5, features $(X_p, d_p)_i$ and the pose $\boldsymbol{\xi}_{Wi}$ of each $\mathcal{K}_i$ are processed by monitoring nodes for this purpose. In addition to the current semidense monocular SLAM method, many key frame based SLAM methods using stereo or RGBD sensors could easily be adapted into exploring nodes, given the fact that those methods can produce required data with minimal effort. However, other sensors, such as LiDAR, do not provide the data appearance based features required in monitoring node. Therefore, they require a major change in the monitoring node to function properly.

5. Conclusions

We introduced a distributed monocular SLAM framework that identifies map overlaps based on an appearance based method. Most importantly the framework computes relative transformations of local maps of SLAM nodes, even when their relative starting positions are unknown. We demonstrated empirically that the distributed framework we developed successfully generated the map using multiple agents and localized them in the environment with little amount of error. We achieved a pose location RMSE between 0.0097 m and 0.0136 m for experiments that were conducted in a 1 m × 1.5 m scene. Each node traveled about 4 m on average. We developed an empirical set up that generated the data set with associated ground truth to be used for extensive evaluation and validation of the distributed SLAM method. This data set has been made publicly available for other researchers to use. We also developed an augmented

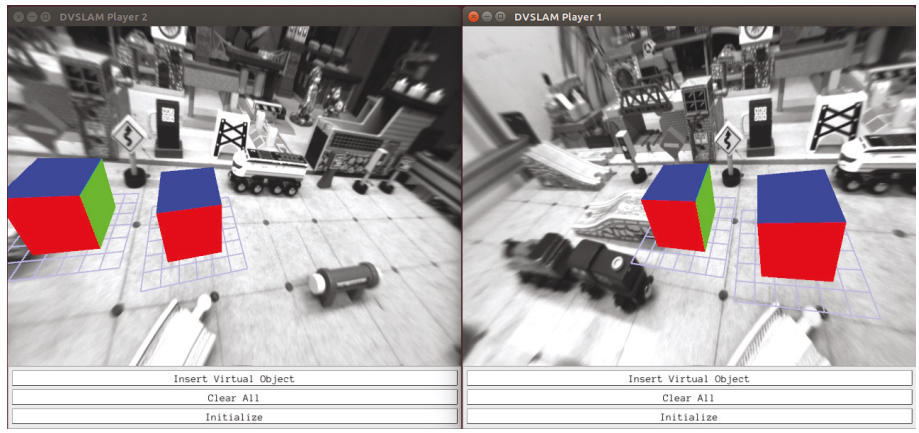


Figure 12: Same set of virtual objects is viewed from two different exploring nodes.

reality application to showcase how two nodes can use our framework to interact with the shared global map.

Our next steps would be to improve exploring node's SLAM process by incorporating features in pose graph optimization. That would help greatly in supporting public datasets as well. Furthermore, we will evaluate ORB descriptors instead of SIFT descriptors to improve performance and reduce the network bandwidth usage.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

References

[1] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles*, I. J. Cox and G. T. Wilfong, Eds., pp. 167–193, Springer-Verlag, New York, NY, USA, 1990.

[2] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Proceedings of the the AAAI National Conference on Artificial Intelligence (AAAI '02)*, pp. 593–598, 2002.

[3] M. Montemerlo, S. Thrun, D. Roller, and B. Wegbreit, "Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '03)*, pp. 1151–1156, August 2003.

[4] C. Cain and A. Leonessa, "FastSLAM Using Compressed Occupancy Grids," *Journal of Sensors*, vol. 2016, Article ID 3891865, 2016.

[5] F.-J. Pei, H.-Y. Li, and Y.-H. Cheng, "An improved FastSLAM system based on distributed structure for autonomous robot navigation," *Journal of Sensors*, vol. 2014, Article ID 456289, 9 pages, 2014.

[6] R. Martinez-Cantin and J. A. Castellanos, "Unscented SLAM for large-scale outdoor environments," in *Proceedings of the IEEE IRS/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, pp. 3427–3432, Edmonton, Canada, August 2005.

[7] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[8] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR '07)*, pp. 225–234, Nara, Japan, November 2007.

[9] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the Association for Computing Machinery*, vol. 24, no. 6, pp. 381–395, 1981.

[10] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, Cambridge University Press, 2nd edition, 2004.

[11] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Undle Adjustment—A Modern Synthesis," in *Proceedings of the Vision algorithms: theory and practice*, vol. 2000 of *Lecture Notes in Computer Science*, pp. 298–372, Springer Berlin Heidelberg, Corfu, Greece.

[12] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Real-time monocular SLAM: why filter?" in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '10)*, pp. 2657–2664, IEEE, May 2010.

[13] H. Strasdat, J. M. M. Montiel, and A. Davison, "Scale drift-aware large scale monocular slam," in *Proceedings of the Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[14] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *Proceedings of the 2011 IEEE International Conference on Computer Vision, ICCV 2011*, pp. 2320–2327, esp, November 2011.

[15] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct monocular SLAM," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8690, no. 2, pp. 834–849, 2014.

[16] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proceedings of the 14th IEEE International Conference on Computer Vision (ICCV '13)*, pp. 1449–1456, Sydney, Australia, December 2013.

[17] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular SLAM," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009.

[18] E. Nettleton, S. Thrun, H. Durrant-Whyte, and S. Sukkarieh, "Decentralised slam with low-bandwidth communication for teams of vehicles," in *Proceedings of the Field and Service Robotics Conference (FSR '06)*, vol. 24, pp. 179–188, 2006.

[19] L. Paull, G. Huang, M. Seto, and J. J. Leonard, "Communication-constrained multi-AUV cooperative SLAM," in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation, ICRA 2015*, pp. 509–516, usa, May 2015.

[20] A. Howard, L. E. Parker, and G. S. Sukhatme, "The SDR experience: Experiments with a large-scale heterogeneous mobile robot team," *Springer Tracts in Advanced Robotics*, vol. 21, pp. 121–130, 2006.

[21] F. Dieter, K. Jonathan, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1338, 2006.

[22] D. Zou and P. Tan, "CoSLAM: Collaborative visual SLAM in dynamic environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 2, pp. 354–366, 2013.

[23] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, "Collaborative monocular SLAM with multiple Micro Aerial Vehicles," in *Proceedings of the 2013 26th IEEE/RSJ International Conference on Intelligent Robots and Systems: New Horizon, IROS 2013*, pp. 3963–3970, jpn, November 2013.

[24] J. Solà, A. Monin, M. Devy, and T. Vidal-Calleja, "Fusing monocular information in multicamera SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 958–968, 2008.

[25] S. Bernard Williams, *Efficient solutions to autonomous mapping and navigation problems [Ph.D. thesis]*, The University of Sydney, New South Wales, Australia, 2001.

[26] S. B. Williams, G. Dissanayake, and H. Durrant-Whyte, "Towards multi-vehicle simultaneous localisation and mapping," in *Proceedings of the Robotics and Automation, IEEE International Conference (ICRA '02)*, pp. 2743–2748, usa, May 2002.

[27] H. Kin and P. Newman, "Multiple map intersection detection using visual appearance," in *Proceedings of the International conference on computational intelligence, robotics and autonomous systems (CIRAS' 05)*, 2005.

[28] A. Cunningham, M. Paluri, and F. Dellaert, "DDF-SAM: Fully distributed SLAM using constrained factor graphs," in *Proceedings of the 23rd IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010*, pp. 3025–3030, twn, October 2010.

[29] A. Cunningham, V. Indelman, and F. Dellaert, "DDF-SAM 2.0: Consistent distributed smoothing and mapping," in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation, ICRA 2013*, pp. 5220–5227, deu, May 2013.

[30] E. Montijano, R. Aragues, and C. Sagüés, "Distributed data association in robotic networks with cameras and limited communications," *IEEE Transactions on Robotics*, vol. 29, no. 6, pp. 1408–1423, 2013.

[31] R. Gamage and M. Tuceryan, "An experimental distributed framework for distributed Simultaneous Localization and Mapping," in *Proceedings of the 2016 IEEE International Conference on Electro Information Technology, EIT 2016*, pp. 665–667, usa, May 2016.

[32] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[33] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[34] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proceedings of the International Conference on Computer Vision Theory and Application (VISSAPP '09)*, pp. 331–340, Lisboa, Portugal, Feburary 2009.

[35] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Journal of the Optical Society of America A*, vol. 4, no. 4, pp. 629–642, 1987.

[36] M. Quigley, K. Conley, B. Gerkey et al., "ROS: An Open-Source Robot Operating System," in *Proceedings of the ICRA workshop on open source software (ICRA '09)*, vol. 3, 5 pages.

[37] M. Burri, J. Nikolic, P. Gohl et al., "The EuRoC micro aerial vehicle datasets," *International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[38] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 3354–3361, Providence, RI, USA, June 2012.

[39] J. Engel, V. Usenko, and D. Cremers, "A Photometrically Calibrated Benchmark For Monocular Visual Odometry," 2016, https://arxiv.org/abs/1607.02555.

[40] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proceedings of the 25th IEEE/RSJ International Conference on Robotics and Intelligent Systems (IROS '12)*, pp. 573–580, October 2012.