# A collaborative Augmented Reality framework based on Distributed Visual SLAM

Ruwan Egodagamage* and Mihran Tuceryan†
Department of Computer and Information Science
Indiana University - Purdue University - Indianapolis
Indianapolis, Indiana 46202
Email: rjegodag@iupui.edu* and tuceryan@iu.edu†

***Abstract*—Visual Simultaneous Localization and Mapping (SLAM) has been used for markerless tracking in augmented reality applications. Distributed SLAM helps multiple agents to collaboratively explore and build a global map of the environment while estimating their locations in it. One of the main challenges in Distributed SLAM is to identify local map overlaps of these agents, especially when their initial relative positions are not known. We developed a collaborative AR framework with freely moving agents having no knowledge of their initial relative positions. Each agent in our framework uses a camera as the only input device for its SLAM process. Furthermore, the framework identifies map overlaps of agents using an appearance-based method.**

## I. Introduction

Markerless tracking has been a goal of many augmented reality applications, and the Simultaneous Localization and Mapping (SLAM) has been a robust framework to accomplish this. The robotics community defines the SLAM problem as an agent creating a map of an unknown environment using sensors while localizing itself in it. To localize the agent properly, an accurate map is required. To generate an accurate map, localization has to be done properly. Which means this localization and mapping need to be done simultaneously to benefit each other.

Inexpensive, ubiquitous mobile agents with cameras and image processing tools made them a popular choice of a sensor for SLAM. Most Visual SLAM approaches relied on detecting features and generating sparse maps using them. More recent solutions with direct featureless methods [1] generate semi-dense maps of the environment. Dense maps provide many benefits over sparse maps including, better agent interactions with the environment or objects, better scene interaction for augmented reality applications, and better object recognition with enhanced data. However, in practice, direct featureless methods require significant overlaps between key frames, with narrower baselines. This adds a limit to the movement of the camera. Furthermore, direct method alone could not handle large loop closures.

Many researchers investigated on how to use multiple agents to perform SLAM: called collaborative or distributed SLAM. Distributed SLAM increases the robustness of SLAM process and makes it less vulnerable to catastrophic failures. Challenges in distributed SLAM are computing map overlaps and sharing information between agents with limited communication bandwidth.

In our proposed framework, agents generate a local semi-dense map utilizing direct featureless SLAM approach. The framework also extracts features and uses them to detect loop closure in local maps and to compute map overlaps between agents. Agents do not use any prior knowledge of their initial poses to determine map overlaps. We show that the framework can be used in collaborative AR applications.

## II. Related Work

In a seminal paper, Smith et al. [2] introduced an Extended Kalman Filter (EKF) based solution for the SLAM problem (EKF-SLAM). The EKF incrementally estimates the posterior distribution over agent pose and landmark positions. The covariance matrix grows with the number of landmarks. Even a single landmark observation leads to an update of the covariance matrix, limiting the number of landmarks EKF-SLAM could handle due to the excessive computational cost. Furthermore, EKF-SLAM has Gaussian noise assumptions. A Monte Carlo Sampling (particle filter) based approach by Montemerlo et al. [3] named FastSLAM, addressed above limitations and supported non-linear process models and non-Gaussian pose distributions.

Davison et al. [4] introduced Monocular Visual SLAM (MonoSLAM); a method of capturing the path of a freely moving camera while generating a sparse map. The generated sparse map consisted of image patches as features. They combined EKF-SLAM and Particle Filtering (PF) for estimation and feature initialization respectively. Klein et al. in [5] 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 keyframes. Similar to MonoSLAM, a set of sparse point features represented the map. RANSAC [6] and 5 point algorithm [7] initialized the system. A global Bundle Adjustment (BA) [8] with Levenberg-Marquardt optimization [7] adjusted the pose of all keyframes. Furthermore, a local

This is the author's manuscript of the article published in final edited form as:
Egodagamage, R., & Tuceryan, M. (2017). A Collaborative Augmented Reality Framework Based on Distributed Visual Slam. In 2017 International Conference on Cyberworlds (CW) (pp. 25–32). https://doi.org/10.1109/CW.2017.47

BA changed the pose of a subset of keyframes allowing a reasonable rate of exploration.

BA worked 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's analysis in [9] showed increasing the number of image features acquired per frame was more beneficial than incorporating information from increased number of closely placed camera frames. They argued that the former increases the accuracy of the motion estimation and a better map estimation for a given computational budget. Their analysis hence favored 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.

The work by DTAM by Newcombe et al. [10] and LSD-SLAM by Engel et al. [1] utilized image pixel intensities directly instead of features for SLAM. Their systems generated dense or semi-dense maps of the environment. Furthermore, these direct methods were more robust to motion blur of images.

### A. Distributed SLAM

A naive brute-force method could communicate all sensor observations and map updates between agents in a distributed SLAM system. However, computational resources and communication bandwidth of an agent are limited. Furthermore, the distributed network is subject to failures of nodes and links. Therefore, to overcome these challenges, a proper and intelligent approach is required for a distributed SLAM system.

If agents know either their relative locations or map overlaps they can easily generate a unique, globally consistent map. For example, in [11], relative locations of the agents were provided by global positioning sensors (GPS). It was also relatively easier to determine map overlaps if the relative initial poses of all agents are known. For example, Paull et al. in [12] initialized agents with known GPS location information.

The problem becomes difficult if the relative locations of agents are unknown. In some contributions, agents continued building local sub-maps until they meet each other. Howard et al. [13] proposed a method where each agent could detect other agents. The agents use these coincidental encounters to find their relative locations. Dieter Fox et al. in [14] presented a method where each agent actively sought other agents in the environment to find their relative locations.

We used the experimental framework for distributed SLAM that we introduced in [15], for the development of this framework.

## III. System Overview

Our framework consists of two types of distributed nodes; *exploring node* and *monitoring node*. These nodes are deployed on different physical machines and given a globally unique identifier. The framework has one monitoring node and multiple exploring nodes at any given time. The nodes use communication channels to pass messages between each other.

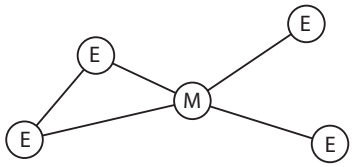



Fig. 1: The network of nodes, exploring nodes (E) are connected to monitoring node (M). Some exploring nodes are connected with each other

We use the Robot Operating System (ROS) [16] infrastructure for our framework. ROS includes *nodes* that are responsible for performing computations. We implemented exploring and monitoring nodes as ROS nodes. ROS also provides named communication busses called *topics* to pass messages between ROS nodes. We use ROS topics as our peer-to-peer communication channels between nodes.

As the name suggests, exploring nodes are responsible for generating a local map of the environment. They periodically send their map to the monitoring node. The monitoring node continuously monitors these map updates to determine potential map overlaps. If it finds an overlap between a pair of exploring nodes, it sends a command to connect those nodes and merge their maps. Figure 1 shows a possible configuration of nodes. As illustrated, exploring nodes are always connected to the monitoring node. If there is a map overlap, two exploring nodes can also be connected to each other. Sections IV and V explain the functionality of exploring node and monitoring node respectively.

We developed a multi-user AR application to demonstrate the collaborative AR potential of our framework. We added an AR window to each exploring node, allowing users to interact in the same environment. This is explained in more detail in section VII.

## IV. Exploring Node

Each exploring node performs semi-dense visual SLAM based on the work by [17]. It uses a single camera as the only input device. It maintains a list of key frames and a pose graph to represent its local map.

### A. Key Frames

The $i^{th}$ **key frame**, $\mathcal{K}_i$ consists of an **absolute pose** $\boldsymbol{\xi}_{Wi} \in \mathbb{R}^7$, an image $I_i$, a map containing $z$ coordinate reciprocals corresponding to non-negligible intensity gradient pixels $D_i$ (an inverse depth map), an inverse depth variance map $V_i$ and a list of features $F_i$. Figure 3 contains a visual representation of $\mathcal{K}_i$ of two key frames. Features of $\mathcal{K}_i$ are computed when we introduce $\mathcal{K}_i$ into the pose graph. In $\mathcal{K}_i$, $i$ corresponds to a 32 bit globally unique identifier. We combine the globally unique node identifier and a locally unique frame identifier to generate a globally unique key frame identifier as shown in Figure 2.

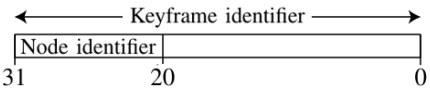



Fig. 2: Globally unique keyframe identifier based on node identifier

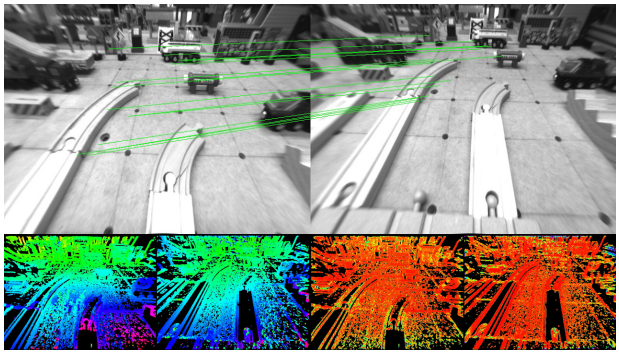


Fig. 3: We show the matched features between key frames $\mathcal{K}_i$ and $\mathcal{K}_j$ superimposed on the images $I_i$ and $I_j$ (top). We also show the pseudo-color encoded $D_i$ and $D_j$ (bottom left) and pseudo-color encoded $V_i$ and $V_j$ (bottom right).

### B. Pose Graph

Pose graph edges $\varepsilon_{ji}$ contain similarity transformations $\boldsymbol{\xi}_{ji}$ and $\boldsymbol{\Sigma}_{ji}$ constraints. Here $\boldsymbol{\xi}_{ji} \in \mathbb{R}^7$, $\boldsymbol{\Sigma}_{ji}$ are relative pose transformations, and corresponding covariance matrix between $i^{th}$ and $j^{th}$ the key frames respectively. Both absolute pose $\boldsymbol{\xi}_{Wi}$ and similarity transformation $\boldsymbol{\xi}_{ji}$ are encoded with a translation (three components) and orientation with scale using a quaternion (four components).

### C. SLAM Process

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 [1].

### D. Features

We used SURF [18] features and SIFT [19] descriptors in our framework. Our choice did not adversely affect on the real-time performance, given we only compute features in key frames.

Next we filter features so that, the $p$th feature in $\mathcal{K}_i$ satisfies,

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

where $X_p$ represents feature location.

For every salient feature in $F_i$, the corresponding 3D location $X_p$ and the descriptor $d_p$ are computed.

### E. Communication with the Monitoring node

Between exploring and monitoring nodes, there are three communication channels. Exploring node sends its new key frame $\mathcal{K}_i$ along with features $F_i$ through the *key frames channel*. After every pose graph optimization, the pose graph is sent through *pose graph channel*. Exploring nodes receive commands through *commands channel*.

Upon receiving a loop closure command from Monitoring node with $\boldsymbol{\xi}_{ji}$, the exploring node checks whether there is an existing edge $\varepsilon_{ji}$ between $\mathcal{K}_i$ and $\mathcal{K}_j$ vertices of the pose graph. If an existing edge is found, it discards the loop closure command. Otherwise, it inserts the new edge and completes the process by performing another iteration of pose graph optimization.

### F. Communication with other Exploring nodes

As shown in Figure 1, two overlapping exploring nodes can communicate with each other. Map overlap key frame correspondences are provided by the Monitoring node. Once the connection is made, each exploring node sends its map to its counterpart through *map merge channel*. Once the map is received, the key frame correspondences are directly transformed into new constraints between pose graphs of $e_i$ and $e_j$.

Figure 4 shows how $e_i$ and $e_j$ were generating their own maps before merging. Right hand side map of Figure 5 shows a resulting merged map two exploring nodes. 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.

### G. Modules of the exploring node

Figure 6 shows modules 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.

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.

## V. Monitoring Node

Exploring nodes of our distributed framework do not know their relative poses at the beginning. Monitoring Node's Map

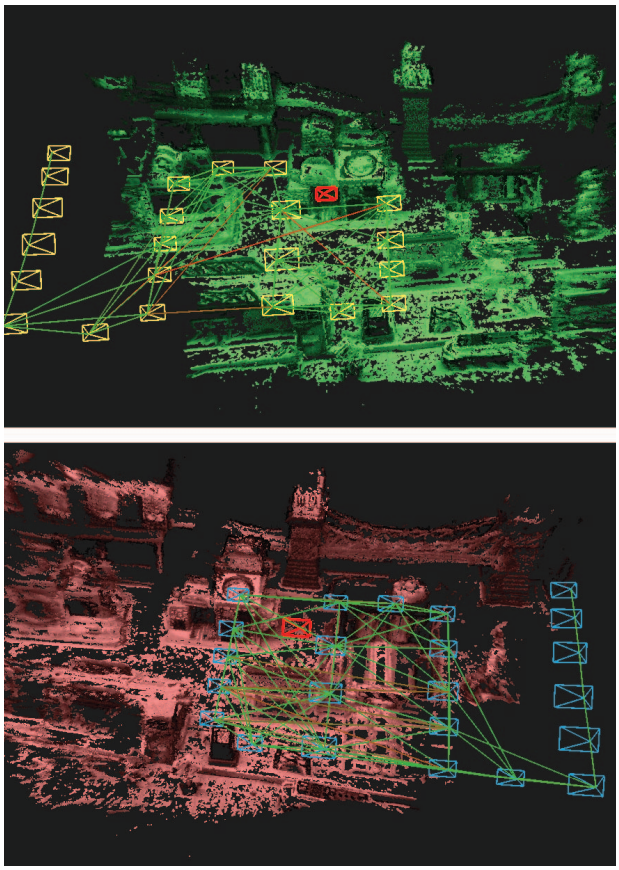


Fig. 4: Map generation process of two exploring nodes. Each exploring node has its own coordinate system. Relative transformations between coordinate systems are initially not known.

overlap detection module is responsible for detecting and computing corresponding relative pose between nodes. It also detects loop closure of each exploring node.

Monitoring node maintains an $N$ number of key frame databases $DB_i$. Here $N$ equals to the number of exploring nodes in the framework. All incoming key frames $\mathcal{K}_i$, are matched against all these key frame databases. The matching takes place in parallel in $M$ number of threads. The number $M$ ($< N$) is configured based on available system resources.

### A. Key frame database

Each key frame database consists of key frames of one exploring node. Each incoming key frame $\mathcal{K}_i$ is matched against the entries in the database using FLANN[20] feature matching method. If there are more than 10 number of matches with another key frame $\mathcal{K}_j$, it is concluded that there is an overlap between key frames $\mathcal{K}_i$ and $\mathcal{K}_j$. If these key frames belong to same exploring node, a loop closure, is found. Otherwise, the result is submitted to the *Fusion Graph*.

*1) Fusion graph:* All available exploring nodes are represented as vertices in the fusion graph as shown in Figure 7. 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 $i^{th}$ exploring node. Then, the fusion graph contains an edge between $e_i$ and $e_j$. The number of features matched between $e_i$ and $e_j$ are represented using $c_{ij}$ as shown in Figure 7. Note that the edge between $e_i$ and $e_j$ could represent matching features between many key frame pairs.

Assume that the fusion graph edge having the largest $c_{ij}$ satisfies,

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

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, we compute the rigid body transformation between $e_i$ and $e_j$, $\boldsymbol{\xi}_{ji}$, is computed using a Singular Value Decomposition (SVD) based on the least squares method [21]. $X_p$ of all relevant features between $e_i$ and $e_j$ are used for the computation. The RANSAC algorithm [22] 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 $\mathcal{K}_j$.

*2) Communication with Exploring nodes:* When the monitoring node detects a map overlap between exploring nodes $e_i$ and $e_j$, it issues a merge command through the *commands channel* to both of them. The command contains the relative pose $\boldsymbol{\xi}_{ji}$ between two nodes. The command also contains the map overlap 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 $e_s$. Fusion graph does not look for map overlaps between nodes that are already found overlapping. This prevents issuing merge command to $e_i$ and $e_j$ again.

*3) Modules of the monitoring node:* As shown in Figure 6, the monitoring node has three main modules. The *input stream* module is receiving key frames and pose graphs from exploring nodes. These key frames are submitted to the *map overlap detection* module which processes these key frames against multiple key frame databases in parallel. The *fusion graph* is used to prioritize map merging of exploring nodes.

## VI. Evaluation and Discussion

### A. Public datasets

To evaluate our system, we need a monocular visual SLAM dataset, with multiple trajectories covering a single scene. We considered publicly available datasets, and they did not satisfy our requirements. For example, the dataset EuRoC [23] contains pure rotations which did not work well with the monocular SLAM approach we used. The Kitti [24] 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 [25] dataset does not provide ground truth for all frames and is primarily suitable for evaluating single agent SLAM. Therefore, we

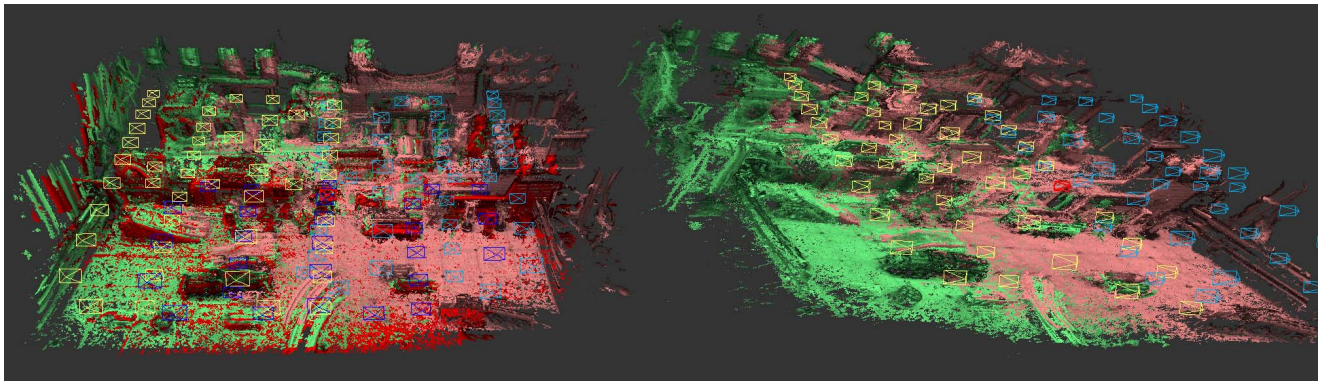


Fig. 5: Resultant maps of two exploration nodes after map merging process. In exploring node on the left, three maps are merged. In exploring node on the right, two maps are merged. It's map and key frames are shown in green and yellow respectively. The map and key frames received from the other node are shown in pink and blue, respectively. Constraints of the pose graph are not shown here to avoid too much clutter in the figure.

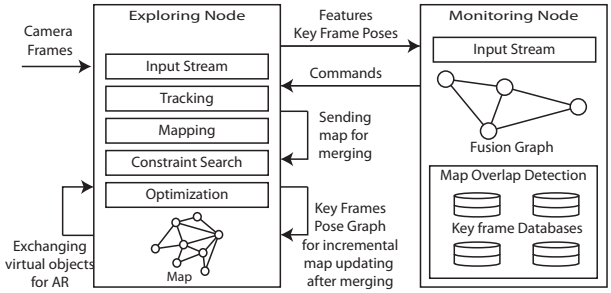



Fig. 6: The distributed framework. In the figure, the arrows looping back to the exploring node rectangle represent communication between two exploring nodes.

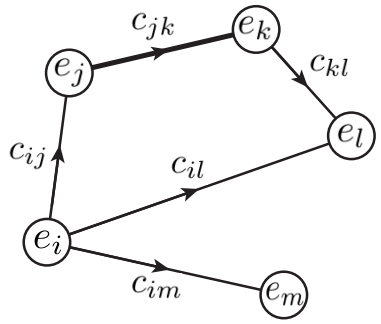



Fig. 7: The fusion graph showing exploring nodes ($e_i$) and the number of matching features ($c_{ij}$) as the weight of each edge. In this example, $c_{jk}$ is higher than other edges (indicated by the thicker edge), so $e_j$ and $e_k$ is merged first. Furthermore, $e_j$'s map is also sent to $e_k$ following the direction of the edge.

created the DIST-Mono dataset to evaluate our system. We also made it publicly available[1].

### B. Experimental setup

Our experimental setup is designed to define the ground truth of a camera motion precisely. As shown in Figure 8 we mounted a Point Grey Firefly MV global shutter camera on a Computer Numeric Controller (CNC) machine. We also prepared a $1m \times 1.5m$ scene containing wooden objects. We then moved the camera along a path roughly four minutes each time, while capturing its location ground truth periodically. We captured $640 \times 480$ resolution camera frames at 60Hz and ground truth at 40Hz. The CNC machine has 0.2mm accuracy in all three axes. We developed an open-source ROS node [2] to capture the ground truth from the TinyG CNC controller.

### C. Dist mono dataset

The dataset consists of five sub-datasets. We defined three camera motion paths, Path A, Path B and Path C. All these paths are on a plane slanted above the scene as shown in Figure 9a. These paths have roughly 10% overlap and three different starting points. We generated two datasets using PathA by rotating the camera around its $z$ axis. In S01-A-0, the camera optical axis and scene Y axis is on a vertical plane. In S01-A-P20, we rotated the camera around its $y$ axis by 20°. This is illustrated in Figure 9b. Similarly, we created datasets S01-B-0, S01-B-N20, and S01-C-0 as shown in Table I.

| Dataset | Path | Initial camera rotation |
|---|---|---|
| S01-A-0 | Path A | 0 |
| S01-A-P20 | Path A | 20 CW |
| S01-B-0 | Path B | 0 |
| S01-B-N20 | Path B | 20 CCW |
| S01-C-0 | Path C | 0 |

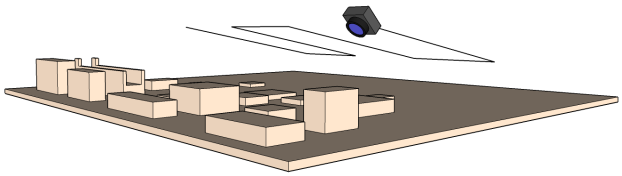
TABLE I: DIST-Mono dataset

### D. Experiments

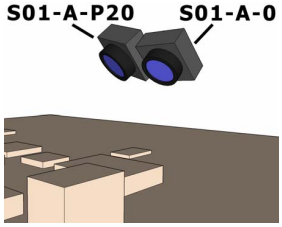
*1) Experiments I:* Two of these datasets were then used to deploy two exploring nodes on two separate physical comput-

[1] http://slam.cs.iupui.edu
[2] http://github.com/japzi/rostinyg


Fig. 8: Experimental setup showing a camera mounted on a CNC machine allowing us to capture ground truth information.Camera mounted on a CNC machine



(a) Motion paths are in a plane slanted above the scene




(b) 20° clockwise rotation

Fig. 9: Camera motion and its initial rotation for datasets

ers. The monitoring node is deployed on a third computer. All these computers run on Ubuntu 14.04 operating system. They are connected via a wired router. This experiment is repeated 100 times, and the resultant transformation between merged two maps is compared against the ground truth.

The resulting relative transformation between dataset S01-A-P20 and dataset S01-B-0 was recorded as shown in Table II (in the table, $\mu$ is the average over 96 trials, and $\sigma$ is the standard deviation). The average error in translation and average error in the rotation were $2.7cm$ and $5.3°$, respectively. Furthermore, it merged maps successfully in 96 out of the 100 attempts. The framework failed to detect map overlaps in the remaining four attempts. Once the framework merged two maps, one exploring node displayed its map as in the right hand side map of Figure 5.

*2) Experiments II:* Similar to the Experiments I, we used dataset SCENE-A-0 and dataset SCENE-B-N20 in two different exploring nodes. After map merging, each exploring node exported its key frame poses in TUM dataset [26] pose format. Most importantly, these poses contain key frames

TABLE II: Relative transformation with rotation ($q$) and translation ($t$)

| $\xi_{ji}$ | Ground truth | Results (96 attempts) | | Average error |
|---|---|---|---|---|
| | | $\mu$ | $\sigma$ | |
| $q_x$ | 0.00 | 0.00 | 0.01 | 5.33° |
| $q_y$ | 0.38 | 0.41 | 0.01 | |
| $q_z$ | 0.05 | 0.08 | 0.01 | |
| $q_w$ | 0.93 | 0.91 | 0.01 | |
| $t_x$(mm) | -680.0 | -706.5 | 6.1 | 27.4 |
| $t_y$(mm) | -70.0 | -74.6 | 17.0 | |
| $t_z$(mm) | 350.0 | 355.8 | 15.0 | |

| 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 |

TABLE III: Experiments and their absolute translation RMSE against ground truth

from both exploring nodes. We then computed the Absolute Translation RMSE [26] against the ground truth. To support the non-deterministic nature of the distributed system, we ran the experiment five times, and the median result is recorded. Similarly, we performed three more experiments with other combinations of datasets as shown in Table III. Given monocular visual SLAM, systems do not capture the scale, we manually calculated that in all experiments to minimize the RMSE error.

Figure 10 shows how estimated key frame poses are compared against ground truth in experiment 3. Red line segments in the figure display the difference between estimated pose location and ground truth location of the key frame.

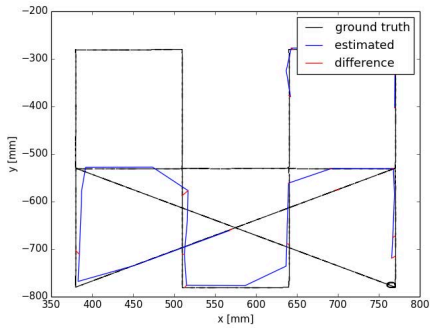
## VII. AR Application

We added an AR window to each exploring node to test our framework. The AR window, allows users to add a virtual object (a simple cube, in our example) into its map. This allows us to demonstrate the collaborative AR potential of the distributed SLAM framework. Each exploring node has its local map so that it can render the augmented scene from its viewpoint. It also knows its pose on the global map. This allows it to render objects added by the other exploring nodes as well. Furthermore, exploring nodes can interact with one another using peer-to-peer communication channels of the framework.

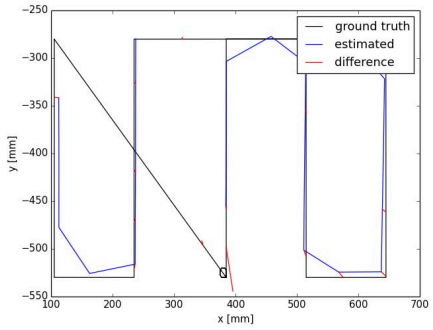
Figure 11 shows AR windows of two exploring nodes and two interactively added cubes.

## VIII. Conclusion

In this paper, we introduced a distributed SLAM framework that identifies map overlaps based on an appearance-based method. The framework operates with no prior knowledge of relative starting poses of its nodes. Using an AR application we have shown that our framework can support collaborative Augmented Reality applications. We also developed a new publicly available dataset and used that for an extensive evaluation of the system.




(a) First exploring node




(b) Second exploring node

Fig. 10: Key frame poses against ground truth

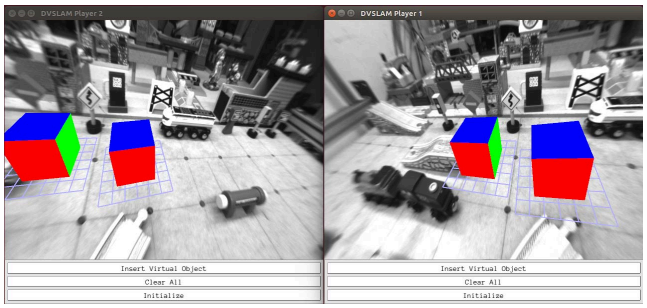



Fig. 11: Same set of virtual objects is viewed from two different exploring nodes

Our next step would be improving the exploring node's SLAM process by incorporating features in pose graph optimization. That would help greatly in supporting public datasets as well. We will also evaluate ORB descriptors instead of SIFT descriptors to improve performance and reduce the network bandwidth usage. Furthermore, we will evaluate the possibility of using a BoW[27] based method instead of the FLANN[20] method we used to detect map overlaps, mainly to improve the performance of the system. The ultimate goal of this framework is to be ported to truly mobile, resource limited platforms and for the computational nodes to run on such mobile devices.

## REFERENCES


[1] J. Engel, T. Schps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Computer Vision ECCV 2014*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2014, vol. 8690, pp. 834–849. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10605-2_54

[2] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles*, I. Cox and G. Wilfong, Eds. Springer New York, 1990, pp. 167–193. [Online]. Available: http://dx.doi.org/10.1007/978-1-4613-8997-2_14

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

[4] A. Davison, I. Reid, N. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 6, pp. 1052–1067, June 2007.

[5] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, Nov 2007, pp. 225–234.

[6] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[7] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[8] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustmenta modern synthesis," in *Vision algorithms: theory and practice*. Springer, 2000, pp. 298–372.

[9] H. Strasdat, J. Montiel, and A. Davison, "Real-time monocular slam: Why filter?" in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 2657–2664.

[10] R. A. Newcombe, S. Lovegrove, and A. Davison, "Dtam: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, Nov 2011, pp. 2320–2327.

[11] E. Nettleton, S. Thrun, H. Durrant-Whyte, and S. Sukkarieh, "Decentralised slam with low-bandwidth communication for teams of vehicles," in *Field and Service Robotics*. Springer, 2006, pp. 179–188.

[12] L. Paull, G. Huang, M. Seto, and J. Leonard, "Communication-constrained multi-auv cooperative slam," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 509–516.

[13] A. Howard, L. Parker, and G. Sukhatme, "The sdr experience: Experiments with a large-scale heterogeneous mobile robot team," in *Experimental Robotics IX*, ser. Springer Tracts in Advanced Robotics, J. Ang, MarceloH. and O. Khatib, Eds. Springer Berlin Heidelberg, 2006, vol. 21, pp. 121–130. [Online]. Available: http://dx.doi.org/10.1007/11552246_12

[14] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, July 2006.

[15] R. Gamage and M. Tuceryan, "An experimental distributed framework for distributed simultaneous localization and mapping," in *2016 IEEE International Conference on Electro Information Technology (EIT)*, May 2016, pp. 0665–0667.

[16] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[17] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Computer Vision (ICCV), 2013 IEEE International Conference on*, Dec 2013, pp. 1449–1456.

[18] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2007.09.014

[19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

[20] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 2009, pp. 331–340.

[21] O. Sorkine-Hornung and M. Rabinovich, "Least-squares rigid motion using svd," 2017, available at https://igl.ethz.ch/projects/ARAP/svd_rot.pdf. [Online]. Available: https://igl.ethz.ch/projects/ARAP/svd_rot.pdf

[22] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[23] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract

[24] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[25] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," in *arXiv:1607.02555*, July 2016.

[26] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[27] D. Gálvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, October 2012.