International Journal of Mechanical Engineering (IJME)
ISSN (P): 2319-2240; ISSN (E): 2319-2259
Vol. 9, Issue 5, Jul–Dec 2020; 1–12

International Academy of Science, Engineering and Technology
Connecting Researchers; Nurturing Innovations

A COOPERATIVE AUGMENTED REALITY (AR) FRAMEWORK BASED ON DISTRIBUTIVE VISUAL SLAM

***Tesfaye Adisu*[1] *& Abaysew Ayele*[2]**

[1]*Research Scholar, Department of Mechatronics Engineering, Near East University, Nicosia / TRNC, Mersin 10 - Turkey*

[2]*Research Scholar, Department of Biotechnology, Shrada University, Greater Noida – 201 310, Uttar Pradesh, India*

ABSTRACT

Distributive Simultaneous Localizations and Mapping (SLAM) helps for multiple agents for exploring and building a global map predicting their locations. The challenge is difficult to identify local map overlaps these agents, especially when their initial relative positions are unknown. So, to address this problem, a collaborative (AR) frame-work with liberally moving agents was used without know-how of their initial comparative positions. Each agent in the framework used a camera only as the input device for its SLAM route.

KEYWORDS: *Augmented Reality; Framework; Nodes; Robotics; Distributive SLAM*

Article History

Received: *22 Oct 2020* **|** ***Revised:*** *29 Oct 2020* **|** ***Accepted:*** *03 Nov 2020*

INTRODUCTION

A visual Simultaneous Localization and Mapping (SLAM) has been using as for marker less tracking during in augmented reality implementations. The term SLAM was formerly developed by Hugh Durrant and John J. Leonard which it's concerned with the applications of building a map of unknown environment by a mobile robot while concurrently navigating the environment using the map, [1]. The robotics community also defined the SLAM problem as an agent of map creator of an unknown site using sensor(s) while concurrently localizing itself in the environment. To localize the agent properly, an accurate map is required. To produce a precise map, self-localization has to been done in appropriate way.

A choice of a sensor for SLAM process is also valuable. Most Visual SLAM approaches relied on detecting features and generating sparse maps using inexpensive, universal mobile agents such as image processing tools and cameras, [2]. Dense maps offer more benefits over sparse maps such like, better agent communications, better object recognition, and better scene interaction for augmented reality applications.

Many researchers explored on how to use multiple agents (distributed SLAM) to perform SLAM. It upsurges the robustness of SLAM process and minimizes disastrous failures. Challenges in distributed SLAM are limited communication bandwidth when sharing information between agents and map's computation overlaps. In this newly proposed framework, agents generate a local quisi-dense map applying direct featureless SLAM method. 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 of their original poses knowledge to determine map overlaps, [3].

LITERATURE VIEWS

SLAM is a procedure by which a robot can build a map of the required environment and concurrently locate itself with respect to the map. Different authors like Smith et al. has been introduced the earliest probabilistic SLAM algorithm, [3]. Extended Kalman (EKF) filter has the weakness of computational complexity, nonlinearity and data association. In large-scale environments, it is difficult to avoid inconsistency [2].And also Smith et al. presented an EKF (Extended Kalman Filter) oriented solution for the SLAM problem, that it incrementally estimates the landmark position and agent pose distribution, [4].Covariance matrix raises with quantity of landmarks. A Monte Carlo Sampling (particle filter) based approach by Montemerlo et al. named Fast SLAM, to address above limitations and supported non-linear process models and non-Gaussian pose distributions, [5].

Davidson et.al. have also presented a Monocular Visual SLAM (Mono SLAM); a method of capturing the path of a liberally moving camera while producing a sparsed map. [6]. EKF-SLAM & Particle (PF) Filtering combined for estimating and featuring initialization. Klein et al. in [6] offered, PTAM (Parallel Tracking and Mapping), which is one of the utmost momentous solutions for visual SLAM. This SLAM solution predominantly focused on accurate & fast mapping in a like environment to Mono SLAM. Its implementations decoupled localizations and mapping, into two threads. The future tracking and front-end thread performs estimation, while the back-end performs mapping and also removing unnecessary key-frames.

Furthermore, Global Bundle Adjustment (GBA) adjusted the pose of entire key frames. BA changed the pose of key frames allowing a reasonable rate of exploration, [7]. GBA worked well for with offline Structure from Motion (SfM).GBA is relatively expensive, although it's recently adopted for monocular visual SLAM solutions. For uniting information, increasing number of image features per frame is more beneficial economically than increasing number of closely placed camera frames, [8]. Moreover, GBA helps to upsurge the number of key features on the map, leading to dense it.

APPROACHES AND METHODS

Distributed SLAM (DSLAM)

In DSLAM, distributed network which is subject to failures of nodes and links, sensor efficacy, computational resources and communication bandwidths could be limited, although are crucial for map updates and initiate intra-communications. To overcome these challenges, a proper and intelligent approach is required for a DSLAM system. If the proportional locations of these agents are provided by the global positioning sensors (GPS) or agents know their locations, they can generate a unique reliable map. It's also comparatively easier to govern map overlaps, if the relative original poses of all agents will be known. However, the problem becomes difficult when the kin locations of agents are unknown. Sometimes, agents continued building local sub-maps until they meet each other, [9].

System Overview

The proposed framework comprises of 2-types of disseminated nodes that deployed on **different machines**; monitoring node and exploring node. The framework has multiple exploring and a monitoring node at a given time. These nodes used for communication to bypass messages amidst each other.

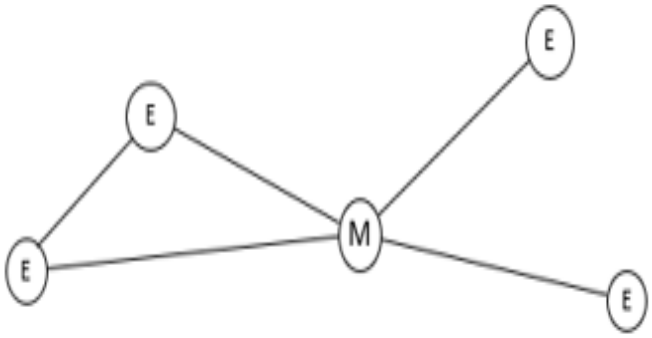


Figure 1: Network of Nodes; Exploring (E) Nodes Connected to a Monitoring (M) Node and Some e-Nodes were Linked to each Other.

E-nodes are accountable for producing a local map of the environment/site and send it periodically to M-node (i.e.it continuously monitors the map's updates to investigate potential map overlaps).If it gets an overlap among two/pair explorer nodes, it sends a command signal to link those nodes and as to merge their maps. As illustrated Figure 1, legally e-nodes are always attached to the monitoring node. If a map overlap occurs, 2-exploringnodes can also be allied to each other. So, in this paper, a poly-user AR application to exhibit the collaborative AR potential of their framework development by different authors has been reviewed. And also an AR window to each exploring node, allowing users to interact in the same environment was added.

Exploring Node

Using a solitary camera as the merely input device, each e-node does semi-dense visual SLAM [10]. It also preserves a list of key-$frames$ and a pose graph to characterize its local map.

- **Key Frames**

The i^{th} **key frame**, k_i consists of an absolute**pose** $\xi_{w_i} \in R^7$, an **image** I_i, a map comprising **z** coordinate reciprocals corresponding to non-negligible intensity gradient pixels $\boldsymbol{D_i}$(an inverse depth map), inverse depth variance map $\boldsymbol{V_i}$ and a list of features $\boldsymbol{F_i}$. Figure 3, below contains a visual representation of $\boldsymbol{K_i}$ of two key frames. Features of Ki are computed when we introduce $\boldsymbol{K_i}$ into the pose graph. In $\boldsymbol{K_i}$, **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.

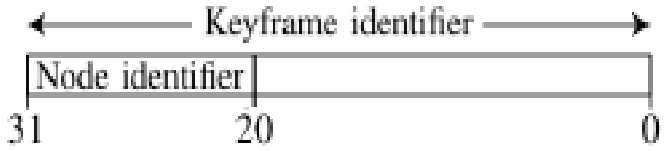


Figure 2: Globally Unique Key Frame Identifier based on Node Identifier.

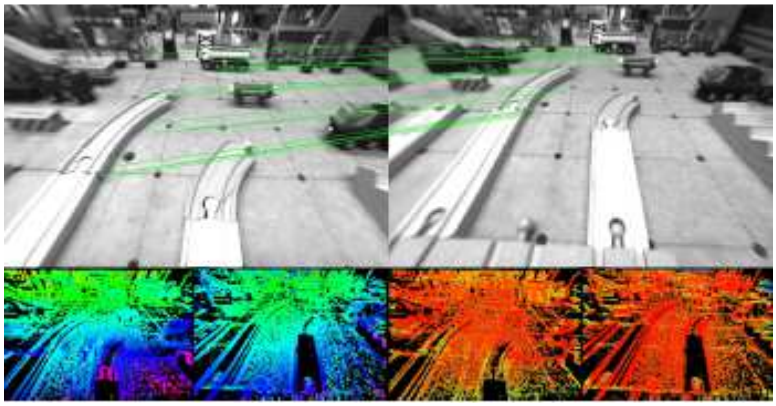


Figure 3

Fig. 3: we matched features b/n key frames $\boldsymbol{K_i}$ and $\mathbf{k_j}$ superimposed on the images Ii and Ij (top). We also show the pseudo color encoded $\mathbf{D_i}$ and $\mathbf{D_j}$ (bottom left) and pseudo color encoded Vi and Vj (bottom right)

- **Pose Graph**

Pose graph edges ε_{ji} contain similarity transformations ξ_{ji} ,and $\sum ji$ constraints. Here, $\xi_{ji} \in R^{7}$, $\sum ji$ are relative pose transformations, and the representing covariance matrix among i^{th} and j^{th} key frames respectively. Both absolute pose ξw_i &likewise transformation ξ_{ji} were programmed with a translation (3-components) and with scale orientation using (4-components).

- **SLAM Process and Features**

The SLAM procedure concurrently tracks the camera alongside the present key-frame K_i and improves its D_i and V_i based on its new observations. Once if this camera meaningfully deviates from the K_i, either a new key-frame is created or/and, if an existing-key frame is selected from the map. Next, if a new key-frame was created, the preceding key-frame used for tracking is implanted into the pose graph. The pose graph is unceasingly optimized in the background [2].In our framework, SURF [11] features and SIFT [12] descriptors are used. Real-time performance, given we only compute features in key frames. So that, the p^{th} feature in K_i key frame, satisfies,

$$V_i(X_p) < T * D_i(X_p)2 \quad (1)$$

Where X_p represents feature location. For every salient feature in *Fi*, the corresponding 3D location X_p and the descriptor *dp* are computed

- **Intra-Communications of Monitoring Nodes and Exploring Nodes**

There are two intra nodes communications; exploring node-to-monitoring and exploring-to-exploring nodes. Between exploring and monitoring nodes, there are three communication channels. E-node sent its new key frame $\boldsymbol{K_i}$ along with features *Fi* through the key-frames' channel. Hereafter, every pose graph optimization, the pose graph is sent through pose graph channel. Exploring nodes receive commands through instructions channel. When receiving a ring closure instruction from M-node with ξ_{ji} , the e-node checked whether there would be an existing edge ξ_{ji} between k_i and $k_j$vertices of the

pose graph. If an existing edge is found, it would discard the loop closure command. Else, it has been inserted the new edge and completed the procedure by doing another iteration of pose graph's optimization.

On the other hand, as displayed in Figure 1 above, the two overlapping e-nodes can link/communicate with each other. Map overlap correspondences are monitored by the M-node. Once the connection is made, each e-node sends its map to its counterpart through map merge channel. Once the map is established, the key-frame correspondences was directly transformed into new constraints between pose graphs of $e_i$ande_j.Fig. 4 shows how e_i and $e_j$before merging; were generating their own maps.

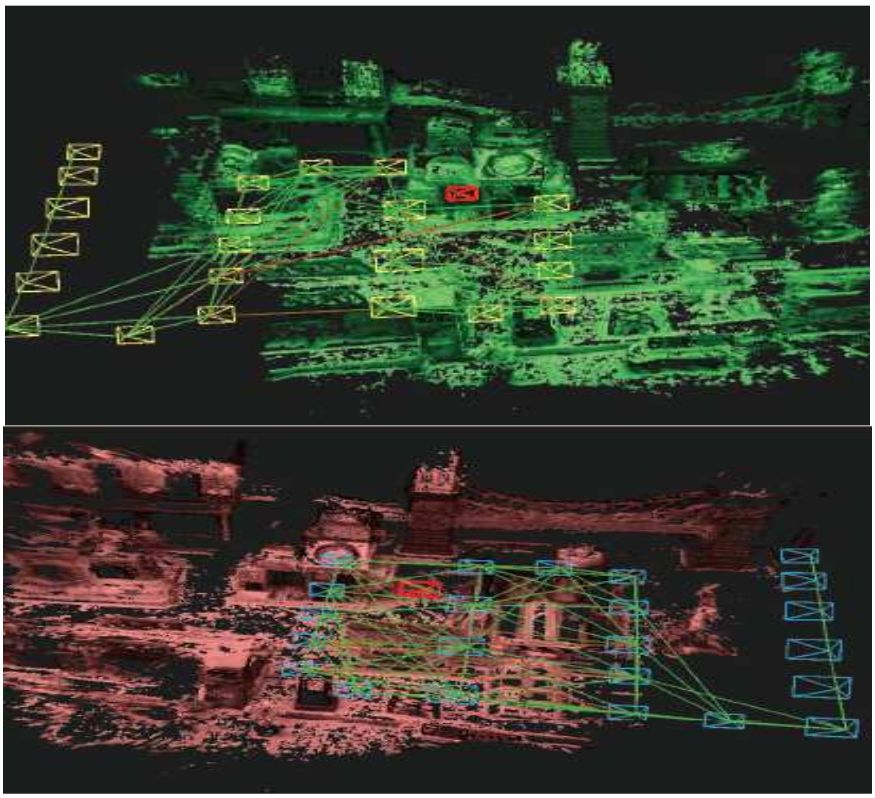


Figure 4: Map Construction Process of Two e-Nodes. Each Exploring Node had its Own Coordinate System.

RHS's map of Figure 5 shows, two e-nodes merged map result. Once merging completed, each e-node listens to its counterpart for new key frames and the pose graph, to increasingly update its map.

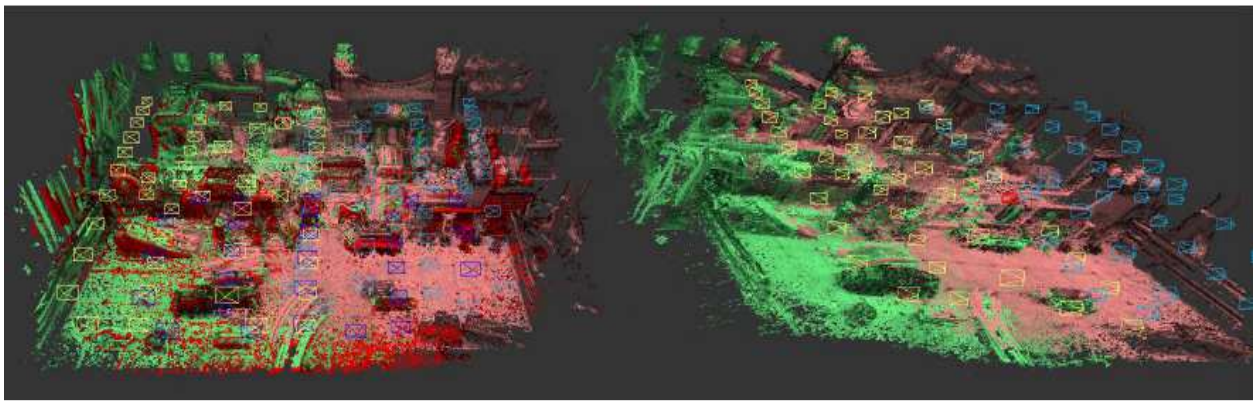


Figure 5

Fig. 5: Resultant maps of two e-nodes after merging procedure. In e-node on the left, three maps are merged. In e-node on the RHS, two maps were merged. It's map and key frames are shown in yellow and green respectively. The maps and key frames delivered from the other node are shown in blue and pink, respectively. Constraints of the pose graph were not displayed here to avoid too much disordered junk in the figure.

- **Modules of Exploring Node**

Figure 6 shows the modules between nodes' communications and the distributed framework. The Exploring node contains of five main modules: tracking, input stream, 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, map, pose graph, and commands. And then all image frames were transferred to the track-module. Pose graph, keyframes, and map transferred to optimization module so that before iterative optimization, they can be merged into map. Commands are treated in the input unit 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 could no longer be applied to track the present frame, a new key frame will be generated. The old key frame can be added up to the map through mapping unit module. The constraint searching module can be used to recover from track failures.

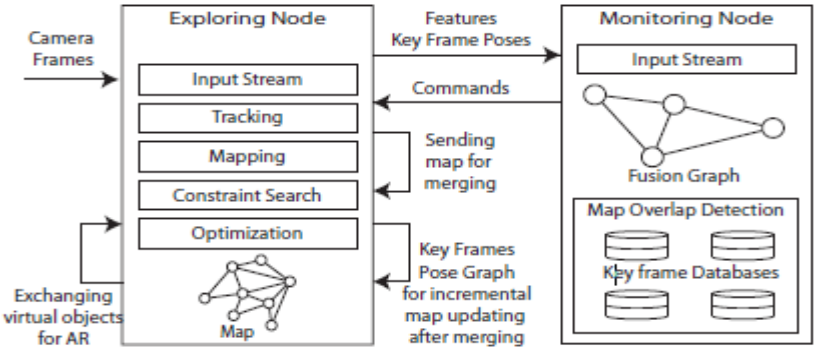


Figure 6

Figure 6 The distributed framework. The arrows led back to the e-node box represent communication between the 2-exploring nodes.

MONITORING NODE

This nodes' map overlap detection/identification 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 $\boldsymbol{N}$ equals to the number of exploring nodes in the framework. All incoming key frames $\boldsymbol{K_i}$, are matched against all these key frame databases. The matching takes place in parallel in $\mathbf{M}$ number of threads. The thread number M (< N) is arranged based on available system resources.

Key Frame Database

Each key frame database entails key frames of 1-exploring node. Each incoming key frame $\boldsymbol{K_i}$ is matched with entries in the database using (fast approximate nearest neighbor) FLANN [13] thru feature matching method. If there are more than 10 number of matches with another keyframe $\boldsymbol{K}$, it is concluded that there is an overlap between keyframes $\boldsymbol{K_i}$ and $\mathbf{k_j}$. If these key frames belong to same e-node, a loop closure is found. Otherwise, the result is submitted to the Fusion Graph.

- **Fusion Graph:** All obtainable e-nodes are represented as vertices in the fusion graph as depicted in Fig. 7 below.

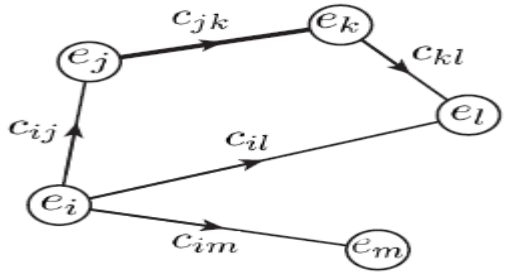


Figure 7

Figure 7: The fusion graph displaying e-nodes (e_i) & 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. Moreover, c_j's map is also sent to $e_k$following direction of the edge.

Assume there is an overlap between key frames Kr and $k_s$and $k_r \in e_i^k$ and $k_s \in e_j^k$, where e_i^krepresent key frames ini^{th}e-node. Then, the fusion-graph comprises an edge amide$_i$&e_j. The number of features coordinated between $e_i$and $e_j$are represented usingc_{ij}as shown in Figure 7. Note that been the edge amide$_i$and $e_j$could symbolize matching features amid many key frame pairs. Assume, the fusion-graph edge having the largest c_{ij}satisfies,

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

While m: an empirical-threshold. Nevertheless, the m-nodes conclude, map overlap avails between e-nodes $e_i$and e_j. Empirically, 120 shared features are found to be a good value for m. The RANSAC algorithm [14] is used to make the computation robust to Outliers. Figure3 indicates a set of matched features between the 2-keyframes, $k_i$and k_j.

- **Communication with Exploring Nodes:** When the m-node detects a map overlay between e-nodes $e_i$and e_j, it concerns a merge order via the commands channel to both of the nodes. The command contains the relative pose ξji between two nodes. Additionally, the command also comprises the map overlap key frame correspondences used to compute the relative pose between e_i&e_j. Likewise, a loop closure instruction was issued to an e-nodee_s, when bothoverlapping key frames k_i and k_j.belong toe_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.

- **Modules of the Monitoring Node:** As in Figure 6,the M-node has 3 main modules. The input stream module is receiving key frames and pose graphs from exploring nodes. These key frames submitted to the map overlay detection, which processes these key frames against multiple key frame's databases. The fusion graph used to order e-nodes for map merging.

RESULT AND DISCUSSION

Experimental Setup

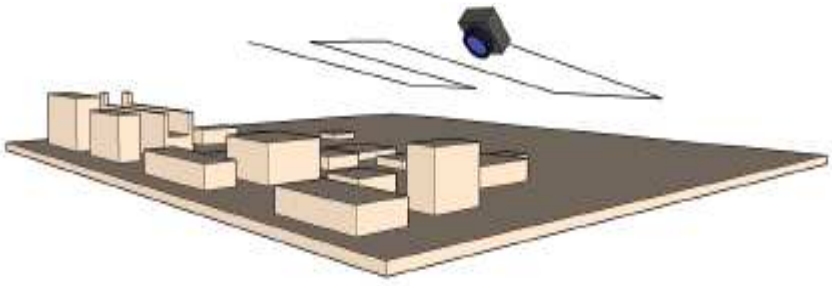
For the new systems setup for distributive SLAM, a monocular visual SLAM dataset is needed, with multiple trajectories covering a single scene. Authors made the DIST-Mono dataset to evaluate our system. Authors' experimental setup was designed to describe the real truth of camera gesture. As shown in Figure 8 researchers have mounted a Point Grey Firefly

MV global shutter camera on a Computer Numeric Controller (CNC) machine. A 1m × 1.5m scene containing wooden objects was also prepared. And the camera was moved along a path roughly 4 minutes each time, while capturing periodically its location ground truth. 640×480 resolution camera frames was also captured at 60Hz andground truth at 40Hz. The CNC Machine has 0.2mm accuracy in all 3-axes. An open-source ROS node [http://github.com/japzi/rostinyg]was also developed in this case to capture the ground truth from the TinyG CNC controller.

Figure 8: Experimental Arrangement Viewing a Camera Straddled on a CNC Machine Permitting us to Capture Real Information.

Dist Mono Dataset

The dataset contains of 5 sub-datasets. Three camera motion paths were defined, Path-A, Path-B & Path-C. All these paths were on a plane inclined above the scene as depicted in Fi-9a. These paths have roughly 10% overlay and 3 dissimilar starting points. Two datasets using Path-A, were generated by rotating the camera around its z-axis. In S01-B-0, the camera scene Y-axis and optical axis was on a vertical plane. In S01-B-P20, the researcher rotated the camera about its y-axis by 20◦ which is demonstrated in Fig-9b.



(a) Motion Paths are in the Plane inclined above the Scene.

(b) 20◦ Clockwise Rotation.

Figure 9: Camera Gesture and Its Preliminary turning for Datasets.

Similarly, we created datasets S01-B-0, S01-B-N20, and S01-C-0 as shown in Table 1.

Table 1: DIST-Mono Dataset

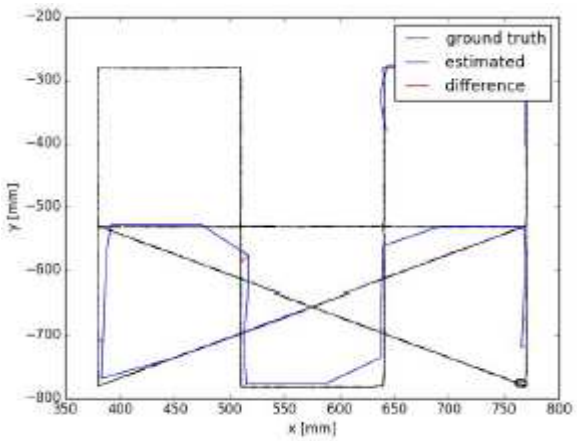
Dataset	Path	Initial camera rotation
S01-A-0	Path A	0
S01-A-P20	Path A	20 CW
S01-B-0	Path B	0
S01-B-N20	Path B	20 CCW
S01-C-0	Path C	0

Experimental Procedures

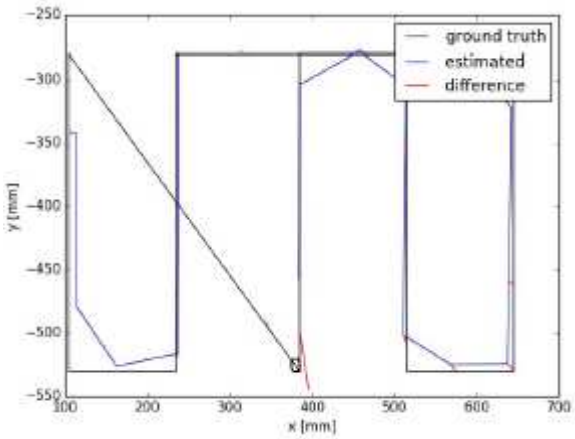
- **Experiments I**: Two of these datasets were then used to deploy two exploring nodes on two separate physical computers. The monitoring node is deployed on a third computer. All these computers run on Ubuntu 14.04 operating system. They were linked via a wired router. This experiment was reoccurred 100 times, and the resultant transform amidst merged 2-maps is compared with the available ground fact. The yielding comparative transformation amidst datasets S01-A-P20 and S01-B-0 was recorded as depicted in Table II (in this table, μ was the average of 96 subsequent trials, and σ is the standard deviation). The average error in translation and average error in the rotation were 2.7cm and 5.3◦, respectively. Moreover, it merged/combined maps in successful way in 96 trialsout of the 100 repetitive attempts. The framework been unsuccessful to detect map overlaps only in the remaining 4-attempts. Once the framework merged2-maps; one e-node displayed its map as in the right-hand side map of Fig-5.

- **Experiments II**: Alike Experiments I, the researcher used dataset SCENE-A-0 and dataset SCENE-B-N20 in 2 unlike e-nodes. After merging of map, each e-node exported its keyframe's poses in TUM dataset [26] pose format. Most importantly, these poses comprise keyframesfrom both exploring nodes. Absolute Translation RMSE [26] was computed against the ground truth. To support the non-deterministic landscape of the distributed system, here the researchers has run experiment for 5-times, &the median outcomewas recorded. In the same way, they performed 3-extra experiments with other dataset's combinations as depicted in Table-III. Given monocular visual SLAM, systems do not capture the scale, then, they have manually calculated to minimize theRMSE errorin all experiments.

Figure 10 reveal how estimated key frame poses were compared viz. the ground reality in experiment-3. Red line segments in the figure reveal the difference between estimated pose location and ground truth location of the key frame.



(a) First Exploring Node.



(b) Second Exploring Node.

Figure 10: Key Frame Poses against Ground Truth.

AUGMENTED REALITY (AR) APPLICATION

As mentioned in section 3.1, the researchers added AR window to each e-node to test their framework. The AR window, allows users to add a virtual object (a simple cube, in taken example) into its map. This permit them to prove the collaborative AR performance of the distributed SLAM framework. Each e-node has its local map therefore it can condense the augmented scene from its standpoint. It has been also knownits pose on the global map. This allows it to render objects added by the other exploring nodesas well. Moreover, exploring nodes can interact with one another using peer-to-peer communication channels of the framework. Figure 11 displays AR windows of 2-exploring nodes and 2 interactively added cubes.

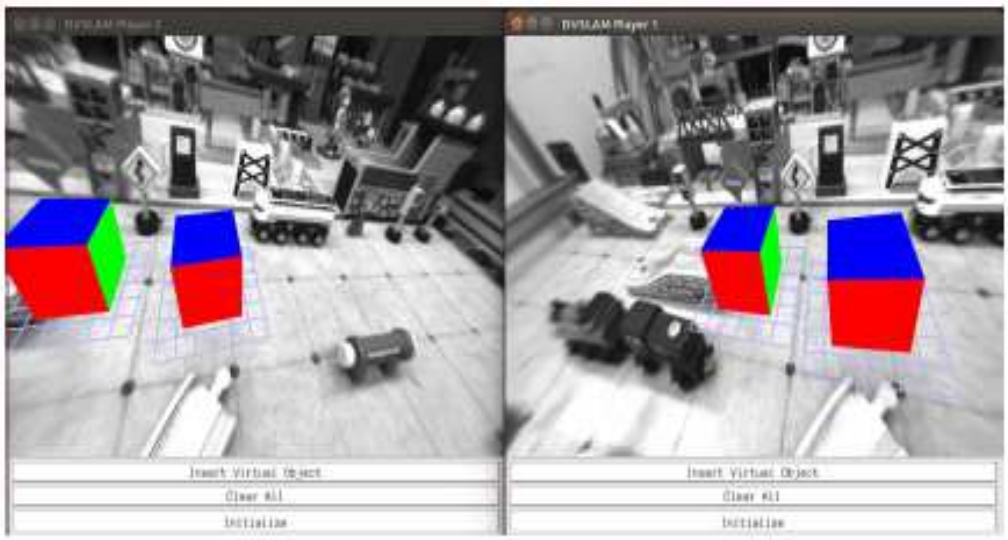


Figure 11: Same Set of Virtual Objects is Viewed from 2 Different Exploring Nodes.

CONCLUSIONS

In this review paper, researchers have familiarized a distributed simultaneous localization and mappingoutline that has been recognizing map overlaps grounded on an appearance-based method. The framework operated with no prior know-how of relative starting poses of its nodes. Via the AR application, they have been shown that their framework can support collaborative Augmented Reality applications. The researchers also have developed a new publicly accessible dataset and

used that for an extensive evaluation of the entire system. Their next step would be improving the exploring node's SLAM process by integrating features in pose graph optimization, which would also help critically in supporting public datasets as well. ORB descriptors instead of SIFT descriptors to improve performance and reduce the network bandwidth usage would be evaluated. The ultimate goal of this framework is to be ported to truly mobile, resource limited platformsand for the computational nodes to run on such mobile devices.

REFERENCES

1. *Leonard, Durrant-Whyte, International Journal of Robotics Research ''Mobile Robot Localization by Tracking Geometric Beacons'',1992.*
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. *Ruwan. E and Mihran. T, ''A Collaborative Augmented Reality Framework Based on Distributed Visual Slam,'' Conference Paper · Sept. 2017.*
4. *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.*
5. *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.*
6. *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.*
7. *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.*
8. *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.*
9. *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.*
10. *H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speededup 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*
11. *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*
12. *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.*

13. *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*

14. *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.*