Non-overlapping RGB-D Camera Network Calibration with Monocular Visual Odometry

Kenji Koide[1] and Emanuele Menegatti[2]

***Abstract*— This paper describes a calibration method for RGB-D camera networks consisting of not only static overlapping, but also dynamic and non-overlapping cameras. The proposed method consists of two steps: online visual odometry-based calibration and depth image-based calibration refinement. It first estimates the transformations between overlapping cameras using fiducial tags, and bridges non-overlapping camera views through visual odometry that runs on a dynamic monocular camera. Parameters such as poses of the static cameras and tags, as well as dynamic camera trajectory, are estimated in the form of the pose graph-based online landmark SLAM. Then, depth-based ICP and floor constraints are added to the pose graph to compensate for the visual odometry error and refine the calibration result. The proposed method is validated through evaluation in simulated and real environments, and a person tracking experiment is conducted to demonstrate the data integration of static and dynamic cameras.**

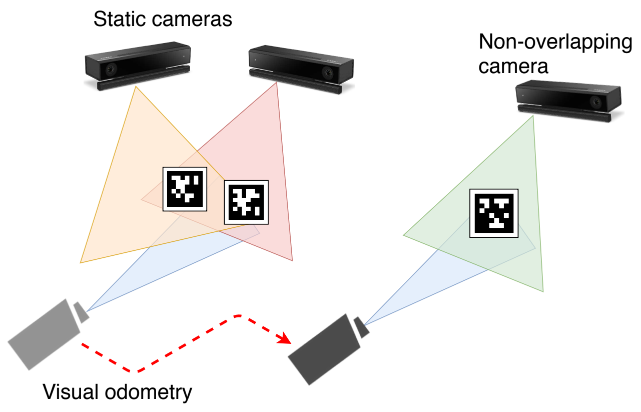


Fig. 1: The non-overlapping cameras are bridged by visual odometry running on a dynamic monocular camera.

I. Introduction

Camera networks have been widely used for various tasks, such as surveillance and monitoring. More recently, the emergence of affordable consumer RGB-D cameras allows the construction of large RGB-D camera network, which realizes rich human-machine interaction capabilities, such as people tracking [1], skeleton tracking [2], and face recognition [3].

Extrinsic camera calibration is one of the essential tasks for camera networks. It is necessary to know the poses of all cameras with respect to a reference frame (so-called "world" frame) to integrate the acquired images. The most common way is to show a calibration pattern (e.g., checkerboard) to two or more cameras, estimate their relative poses, and then show the pattern to another new camera and any of the calibrated ones to estimate the new camera pose with respect to the calibrated ones. This process is repeated until all the cameras are appropriately calibrated. This approach is well-established for overlapping cameras [1], [4].

Such traditional calibration methods, however, have limitations on camera arrangement. For the sake of view area coverage, it is often desirable that cameras have as minimum overlap as possible. In such cases, we need to put the calibration pattern at a small common camera view on image edges. Typically, there is distortion around the periphery of a lens, which may affect the calibration accuracy. Moreover, if camera views are completely separated and have no overlapping regions, camera calibration with the traditional methods will become impossible. Furthermore, the calibration pattern detection becomes difficult when two cameras are distant from each other, leading to deteriorated calibration result due to the degraded pattern detection and pose estimation.

In this paper, we propose a method to calibrate RGB-D camera networks consisting of static non-overlapping cameras and a dynamic monocular camera through a visual odometry technique (See Fig. 1). While calibrating the transformations between overlapping cameras using fiducial tags, we "bridge" non-overlapping camera views using the dynamic camera trajectory estimated by a visual odometry technique. In addition, we introduce a depth image-based calibration refinement technique to make the proposed method robust to visual odometry errors.

The contribution of this paper is three-fold. First, a visual odometry-based camera network calibration method is proposed, which robustly estimates the transformations between non-overlapping static cameras and a dynamic camera. Second, a depth image-based calibration refinement technique is introduced, which effectively improves the calibration accuracy of cameras with small overlapping regions. Third, the implementation is available as an open source software on a public repository [1].

The rest of this paper is organized as follows. Sec. II reviews related work on non-overlapping camera network calibration. Sec. III describes the proposed visual odometry-

[1]Kenji Koide is with the Department of Information Technology and Human Factors, National Institute of Advanced Industrial Science and Technology, Umezono 1-1-1, Tsukuba, 3050061, Ibaraki, Japan, `k.koide@aist.go.jp`

[2]Emanuele Menegatti is with the Department of Information Engineering, University of Padova, via Gradenigo 6/B, 35131, Padova, Italy, `emg@dei.unipd.it`

[1]`https://github.com/koide3/sparse_dynamic_calibration/`

based calibration method and depth image-based refinement technique. Sec. IV evaluates the proposed method in both simulated and real environments, and an experiment to demonstrate the integration of static non-overlapping cameras and a dynamic camera. Sec. V concludes the paper.

II. Related Work

Camera network calibration (i.e., extrinsic calibration of multiple cameras) has been widely studied in the robotics and computer vision communities, and a number of methods have been proposed. The most common and established way is to use a calibration pattern put on an overlapping region of cameras. We can estimate the camera poses with respect to the pattern using a PnP (Perspective-n-Points) algorithm [5]. Moreover, by integrating multiple pattern detection results, such that the sum of the pattern pose errors is minimized [1], [4], we can robustly estimate the transformations between the overlapping cameras. However, for the sake of view area coverage, it is desirable that cameras have minimum or no overlap. In such cases, it is difficult or impossible to calibrate the cameras with the traditional methods.

While overlapping camera network calibration has been well-established, non-overlapping camera network calibration is still a challenging problem [6]. Several works tackled this issue by using moving targets (e.g., pedestrians) [7], [8], [9]. Those works simultaneously estimated the camera poses and the moving target trajectories in regions out of the camera views. However, the uncertainty of the target motion makes the simultaneous estimation difficult, while the calibration accuracy is inherently limited to the topological level.

Huang et al. proposed a calibration method that exploits a mobile robot [10]. They put a checkerboard on the mobile robot and detected it from static cameras. Since the robot motion can be estimated using odometry information even when it moves out of the camera view, it is possible to estimate the pose of a separated camera by detecting the checkerboard when the robot enters in that camera view. However, this method requires accurate odometry information for calibration, as well as a robotic system, which could be more expensive than the camera network itself.

An interesting idea recently proposed is to introduce a support camera that bridges separated camera views. Zhao et al. attached a fiducial tag to each static camera and calibrated the camera poses by observing the tags with a support camera [11]. This method can calibrate the camera poses as long as the support camera can capture the attached tags. However, in case the static cameras are distant or separated by a wall, the support camera fails to capture the tags, and thus, the camera poses cannot be calibrated.

Cansizoglu et al. utilized an RGB-D support camera for non-overlapping camera calibration [12]. They first created a 3D environmental map using an RGB-D SLAM technique, then estimated the static camera poses in the environment by finding the 2D–3D correspondences between the images of the static cameras and the 3D environmental map. Since finding correspondences between an image and a 3D point cloud map is not straightforward, they detected the correspondences by comparing each keyframe image of the RGB-D SLAM with each static camera image using an appearance-based matching technique. Thus, this method requires that any of the keyframes has a view similar to a static camera view.

The method proposed by Pollok and Monari is also based on 2D–3D correspondences [13]. They first performed visual SLAM to obtain the 3D map of the environment, and then, by solving the PnP problem with given correspondences, they estimated the static camera pose with respect to the 3D map. However, in their work, the correspondences are made by hand. Thus, calibration of a large camera network requires significant efforts.

III. Methodology

The proposed camera network calibration method consists of two steps: online visual odometry-based camera network calibration, and offline depth image-based calibration refinement.

We first bridge separated camera views through visual odometry running on a dynamic camera. To estimate the visual odometry scale and the transformations between static cameras and the dynamic camera, we take the pose graph-based landmark SLAM approach with fiducial tags. In this work, we use Apriltag [14] as fiducial landmarks.

In the depth image-based calibration refinement step, we add ICP and floor plane constraints to the pose graph constructed in the online calibration step. ICP constraints allow us to refine the transformation between cameras with small overlap, while floor plane constraints refine the poses of distant and separated cameras. Unlike usual ICP-based approaches, the proposed method does not require large overlap between cameras, since we add these constraints to the pose graph while keeping the graph structure constructed with visual odometry information in the online step.

For the pose graph optimization, we utilize the Levenberg-Marquardt optimizer [15] in g2o [16], a general hyper graph optimization framework.

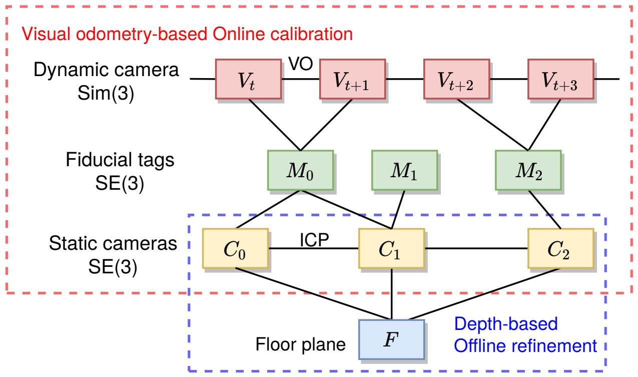


Fig. 2: The proposed graph structure for non-overlapping camera network calibration with visual odometry.

Algorithm 1 Non-overlapping Camera Network Calibration

```
 1: $\mathcal{D} = [v_0, \cdots, v_t, \cdots, v_N]$          ▷ Visual odometry
 2: $\mathcal{C} = [c_0, \cdots, c_M]$                        ▷ Static cameras
 3: $\mathcal{M}^* = [m_i, m_j, \cdots]$                      ▷ Tag detections
 4: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$                ▷ Pose graph
 5: $V_t, C_j, M_i$          ▷ Vertices corresponding to $v_t, c_j, m_i$
 6: procedure CALIBRATION
 7:     for $v_t \in [v_0, \cdots, v_N]$ do                   ▷ Add visual odometry
 8:         $V_t \leftarrow V_{t-1}(v_{t-1}^{-1} \cdot v_t)$   ▷ Initial guess
 9:         $\mathcal{V} \leftarrow \mathcal{V} \cup V_t$
10:         $\mathcal{E} \leftarrow \mathcal{E} \cup \text{Edge}(V_{t-1}, V_t, v_{t-1}^{-1} \cdot v_t)$
11:         for $m_i \in \mathcal{M}_t^v$ do                   ▷ Tags detected at time $t$
12:             ADDTAG$(V_t, m_i)$
13:             $\mathcal{E} \leftarrow \mathcal{E} \cup \text{Edge}(V_t, M_i, m_i)$
14:         Optimize$(\mathcal{G})$
15: procedure ADDTAG$(V_t, m_i)$
16:     if $M_i \in \mathcal{V}$ then
17:         return
18:     $M_i \leftarrow V_t \cdot m_i$                          ▷ Initial guess
19:     $\mathcal{V} \leftarrow \mathcal{V} \cup M_i$           ▷ Add tag vertex $M_i$
20:     for $c_j \in \mathcal{C}$ where $C_j \notin \mathcal{V}$ and $m'_i \in \mathcal{M}^{c_j}$ do
21:         $C_j \leftarrow M_i \cdot m_i'^{-1}$
22:         $\mathcal{V} \leftarrow \mathcal{V} \cup C_j$       ▷ Add cameras which detect $m_i$
23:         $\mathcal{E} \leftarrow \mathcal{E} \cup \text{Edge}(C_j, M_i, m'_i)$
24:         for $m'_k \in \mathcal{M}^{c_j}$ do
25:             ADDTAG$(C_j, m'_k)$ ▷ Add tags detected by $c_j$
```

A. *Online calibration with visual odometry*

The online calibration step takes the form of the graph-based landmark SLAM, in which landmarks are fiducial tags placed in the environment, and odometry is given by a visual odometry technique. The proposed pose graph structure for camera network calibration is illustrated in Fig. 2. V_t indicates a Sim(3) vertex (rotation, translation, and scale) representing the pose of the dynamic camera at time t. V_t is connected with the consecutive vertices V_{t-1} and V_{t+1} based on the dynamic camera ego-motion estimated by visual odometry. C_i and M_j indicate SE(3) vertices (rotation and translation) representing i-th static camera and j-th tag poses. The dynamic and static camera vertices are connected with the tag vertices based on fiducial tag poses detected by each camera. We use only tag detections within a range threshold (e.g., 7.5 m), and apply Huber's robust kernel [17] to each tag detection edge to address the instability of the PnP algorithm for distant tags. See Algorithm 1 for details of the pose graph construction process.

In this work, we use direct sparse odometry (DSO) [18] to estimate the dynamic camera ego-motion. It is worth mentioning that we intentionally made the calibration method independent from the visual odometry algorithm, such that we can choose a different visual odometry algorithm depending on the situation. For instance, in case we use a smartphone as a dynamic camera, we can replace DSO with an IMU-guided visual odometry algorithm, such as [19].

Algorithm 2 Depth Image-based Calibration Refinement

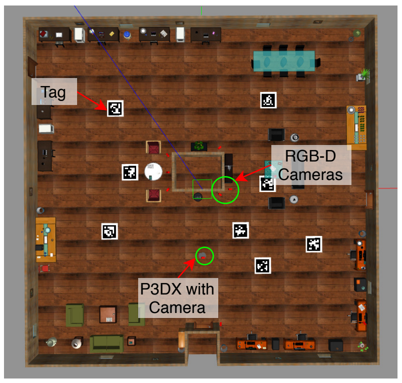
```
 1: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$                ▷ Pose graph
 2: $\mathcal{C} = [c_0, \cdots, c_M]$                        ▷ Static cameras
 3: $\mathcal{P}_i$                                            ▷ Point cloud of $c_i$
 4: procedure REFINEMENT
 5:     $\mathcal{V} \leftarrow \mathcal{V} \cup F$             ▷ Add floor vertex
 6:     for $c_i \in \mathcal{C}$ do                          ▷ Add floor constraints
 7:         $f_i \leftarrow$ DETECTFLOOR$(\mathcal{P}_i)$
 8:         $\mathcal{E} \leftarrow \mathcal{E} \cup \text{Edge}(C_i, F, f_i)$
 9:     for $k \in [0, \cdots, K]$ do                         ▷ ICP iteration
10:         $\mathcal{E}^{\text{ICP}} \leftarrow []$
11:         for $c_i \in \mathcal{C}$ do                      ▷ Find correspondences
12:             for $c_j \in \mathcal{C}$ where $j < i$ do
13:                 $\mathcal{N} = \text{FindCorresponds}(\mathcal{P}_i, \mathcal{P}_j)$
14:                 $\mathcal{E}^{\text{ICP}} \leftarrow \mathcal{E}^{\text{ICP}} \cup \text{Edges}(C_i, C_j, \mathcal{N})$
15:         $\mathcal{E} \leftarrow \mathcal{E} \cup \mathcal{E}^{\text{ICP}}$   ▷ Add ICP constraints
16:         Optimize$(\mathcal{G})$
17:         $\mathcal{E} \leftarrow \mathcal{E} \setminus \mathcal{E}^{\text{ICP}}$  ▷ Remove ICP constraints
```

B. *Offline depth-based calibration refinement*

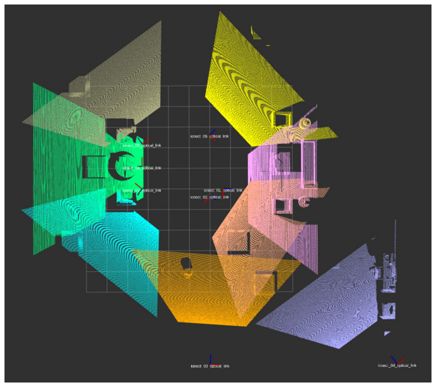
The online calibration step allows us to bridge non-overlapping cameras. However, the calibration accuracy relies on the visual odometry which would get inaccurate due to, for instance, a texture-less environment and quick camera motion. To make the calibration robust to visual odometry errors, we introduce depth image-based floor plane and ICP constraints. Algorithm 2 explains the offline refinement process in detail.

1) Floor plane constraint: We assume that the environment has a single flat floor or parallel floors. We first detect the floor plane from each static camera, then add floor plane constraints, which optimizes the static camera pose such that the detected floor planes become the same or parallel. Following [20], we detect the floor plane from a point cloud using RANSAC [21], and add an edge between the vertices of the static camera and the floor plane. We use a tuple of azimuth angle, elevation angle, intercept length as a minimum plane representation [22]. The floor constraint mode (same/parallel) is selected by a program option; thus, the user has to choose either of them beforehand.

2) ICP constraint: To optimize the transformations between overlapping cameras, we add ICP-based constraints to the pose graph. We extract correspondences between two point clouds using a KD tree-based nearest neighbor search, and validate them with the reciprocal correspondence check. Then, we add edges, which calculate the point-to-plane distance [23] between corresponding points. After adding ICP edges, we optimize the pose graph, remove all the ICP edges, extract new correspondences, and add them to the graph. We repeat this process until the χ^2 error of the pose graph converges.



(a) Simulation environment.


(b) Camera arrangement (*Sim1*).

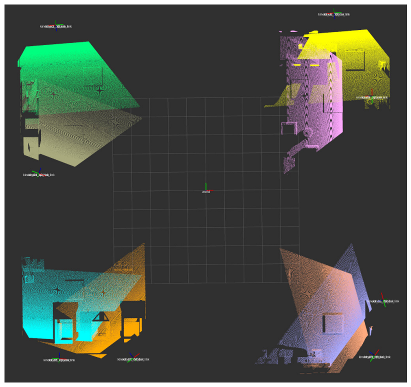

(c) Camera arrangement (*Sim2*).

Fig. 3: The simulation environment.

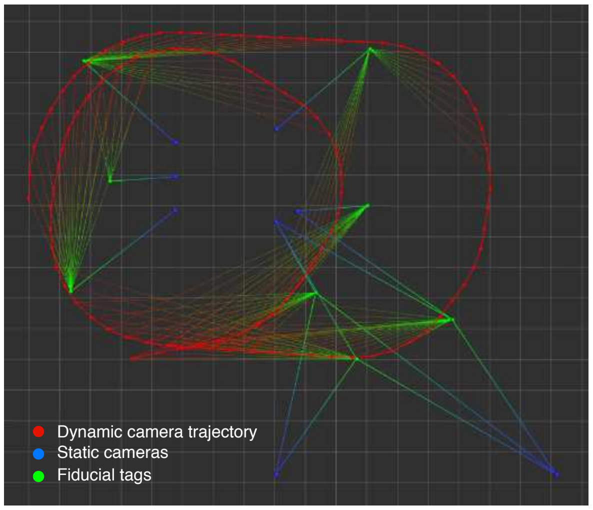


Fig. 4: The constructed pose graph for calibration.

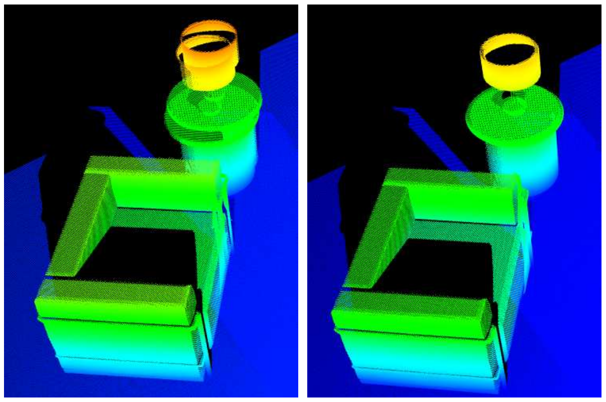

(a) Before refinement.
(b) After refinement.

Fig. 5: The accumulated point cloud before and after the calibration refinement.

IV. Experiments

A. Evaluation in a simulated environment

To evaluate the proposed method, we built a camera network environment with the gazebo simulator [24] and models in 3DGEMS dataset [25]. Fig. 3 (a) shows the simulation environment. We placed eight RGB-D cameras and several Apriltags in the environment so that each camera can observe at least one tag. We tested the proposed method in two camera arrangements referred to as *Sim1* and *Sim2*. In *Sim1*, the cameras have small overlap while being separated into four groups in *Sim2* (See Fig. 3 (b)(c)). For each camera arrangement, we recorded a dynamic camera image stream, in which all the tags are observed at least once using a camera mounted on a mobile robot. To analyze the impact of visual odometry errors on the calibration result, we added small lens distortion (coefficients = 10^{-4}) to the images and evaluated the proposed method with the distorted image stream.

Fig. 4 shows the pose graph constructed in *Sim1* arrangement. The red points indicate the estimated dynamic camera trajectory (the visual odometry scaled and aligned with respect to the world frame). The blue and green points indicate the estimated static camera and tag poses, respectively. The lines between points are relative pose edges. We can see that all the static camera poses are correctly estimated, although they have no common tag detections due to the dynamic camera vertices that bridge the static camera vertices via tag vertices.

Fig. 5 shows the accumulated point clouds of the static cameras before and after the depth-based refinement step. Before the refinement, the point clouds are not well aligned due to the visual odometry error caused by the injected lens distortion. However, after the refinement, ICP and floor plane constraints effectively correct the error, and the point clouds of the cameras are well matched together.

Table I summarizes the calibration errors. In *Sim1*, the visual odometry works well in the case without distortion, and we achieve relatively small calibration errors (1.52 cm and 0.09°, 1.58 cm and 0.17°with and without refinement, respectively) in large environment (15 m in width). In the case with distortion, the calibration error gets almost four times larger (6.10 cm and 0.31°) due to the degraded visual odometry accuracy. However, the depth-based refinement successfully compensates for the visual odometry error, allowing the calibration accuracy to recover back to the level

TABLE I: Calibration accuracy evaluation result in the simulated environment

		w/o lens distortion (small VO error)		w/ lens distortion (large VO error)	
		Translation [m]	Rotation [°]	Translation [m]	Rotation [°]
Sim1	w/o refinement	0.0158 ± 0.0055	0.1704 ± 0.0683	0.0610 ± 0.0408	0.3098 ± 0.2343
	w/ refinement	**0.0152** ± **0.0094**	**0.0918** ± **0.0833**	**0.0158** ± **0.0095**	**0.1337** ± **0.0838**
Sim2	w/o refinement	0.0220 ± 0.0139	0.2805 ± 0.0878	0.0489 ± 0.0237	0.2803 ± 0.0924
	w/ refinement	**0.0151** ± **0.0087**	**0.2275** ± **0.1845**	**0.0342** ± **0.0302**	**0.1349** ± **0.0759**

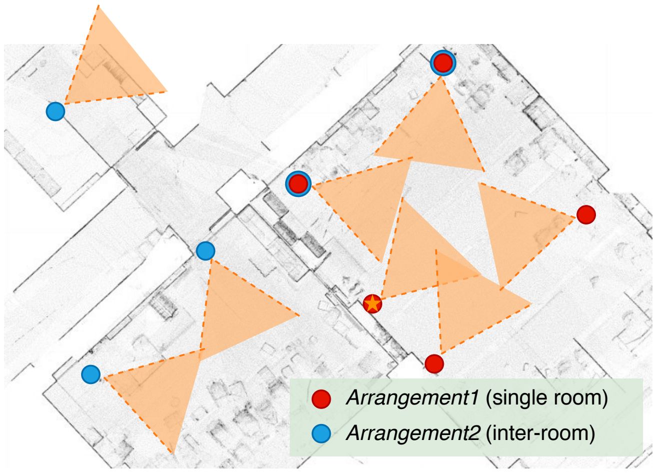


Fig. 6: The evaluation environment. Five Kinects are placed in a single room in *Arrangement1*, while they are placed in multiple rooms in *Arrangement2* such that they are completely separated. The Kinect marked with a star is removed later to reduce the camera overlap.

of the case without distortion (1.58 cm and 0.13°).

In *Sim2*, the separated cameras are properly calibrated owing to the visual odometry-based bridging. Although the calibration accuracy deteriorates slightly (2.20 cm and 0.28°), it is improved after the refinement (1.51 cm and 0.228°). In the case with distortion, the depth-based refinement cannot recover the calibration accuracy to the level without distortion, because it does not create constraints between separated cameras. However, it refines the relative poses between cameras which sharing a common field of view, and thus, the final calibration accuracy improves (from 4.89 cm and 0.28°to 3.42 cm 0.13°). It is worth mentioning that the calibration accuracy can be improved further by observing more fiducial tags with the dynamic camera, so that the visual odometry drift is corrected with multiple observations (i.e., loop closing).

B. *Evaluation in a real environment*

We evaluated the proposed method in a real environment shown in Fig. 6. The number of RGB-D cameras (Kinect V2) is five, and we evaluated the proposed method in two camera arrangements. In *Arrangement1*, we placed all the cameras in a single room such that they have enough overlap. In *Arrangement2*, we placed them in three different rooms to separate their fields of view. We measured the ground truth of the camera positions using a high-definition 3D scanner (FARO FOCUS), recorded an image stream using a mobile

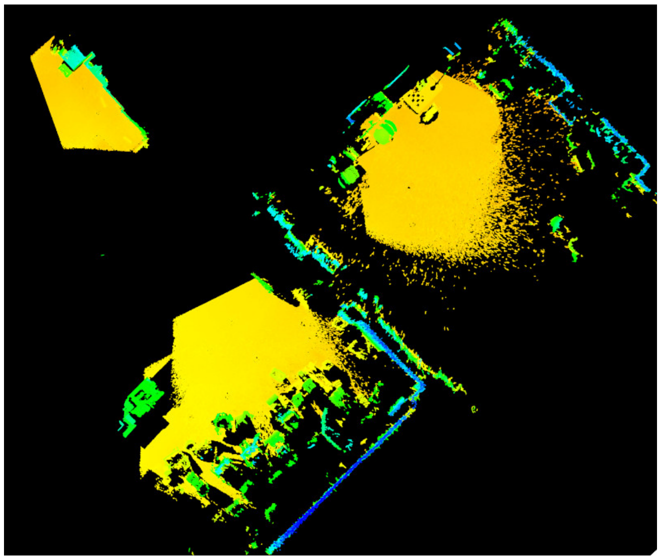


Fig. 7: The accumulated point cloud.

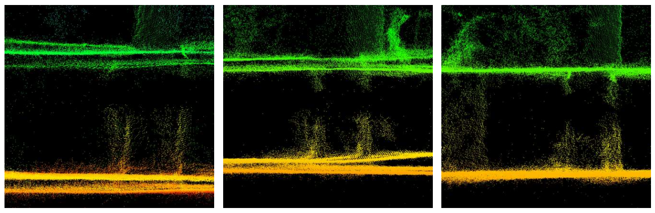

(a) Traditional [1]. (b) Before refinement. (c) After refinement.

Fig. 8: Floor and table surfaces.

camera (Pointgrey Flea3) for each camera arrangement, and calibrated all the camera poses with the proposed method.

In *Arrangement1*, for comparison, we additionally evaluated a traditional camera network calibration method based on checkerboard detection [1]. To determine the impact of the overlap region size on the calibration result, we also removed the camera marked with the star in Fig. 9 to reduce the camera overlap, and calibrated the network with the remaining cameras.

Fig. 7 shows the accumulated point clouds of the cameras calibrated with the proposed method in *Arrangement2*. We can see that all the camera poses are properly estimated, although they are in separated rooms.

Fig. 8 shows the accumulated point clouds of the floor and a table in *Arrangement1* with different calibration methods. Green and orange points correspond to the surfaces of the table and the floor, respectively. The traditional method suffers from distortion and noise on the input images, and as

TABLE II: Calibration accuracy evaluation result in the real environment

	Translation error [m]		
	Traditional	Proposed	
	[1]	w/o refinement	w/ refinement
Arrangement1	0.060 ± 0.047	0.065 ± 0.021	**0.058** ± **0.034**
Arrangement1 (Small overlap)	0.071 ± 0.058	0.064 ± 0.025	**0.060** ± **0.032**
Arrangement2 (Inter-room)	N/A	0.052 ± 0.034	**0.041** ± **0.029**

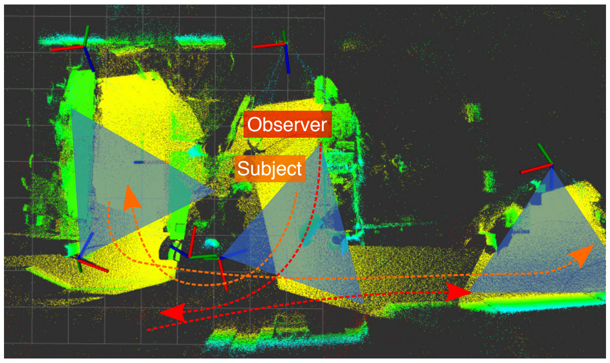


Fig. 9: The experimental environment. A subject walks over separated camera views, and an observer carrying a mobile camera follows them.

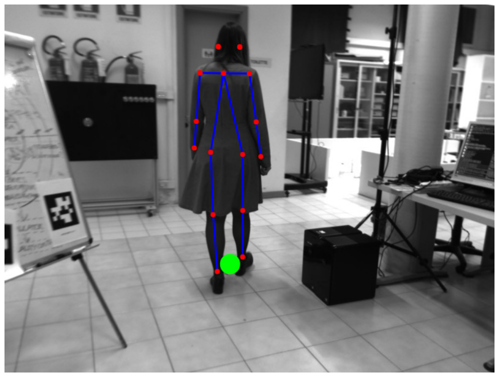


Fig. 10: An example image taken by the mobile camera. The green circle indicates the ankle position to be projected onto the floor plane.

a result, doubled planes are observed. Although the proposed method also suffers from such disturbances, the depth-based refinement enables proper alignment of the point clouds of the static cameras, leading to the observation of consistent surfaces for each floor and table.

Table II summarizes the calibration results. Because it was not feasible to measure the ground truth of the sensor orientations, we evaluated only the RMSE (root mean square erros) of the translation part. We can see that the proposed and the traditional [1] methods show comparable results in *Arrangement1* (about 6 cm errors). When we remove the star-marked camera in *Arrangement1*, the calibration accuracy of the traditional method decreases to 7 cm, because cameras are distant with small overlap, while the accuracy of pattern detection and pose estimation deteriorates in this setting. This result shows that the traditional method requires a large overlap between cameras for accurate calibration, while limiting the camera arrangement. By contrast, the proposed method is not affected by the sparse camera arrangement, because the visual odometry helps to estimate the poses of the distant cameras. In *Arrangement2*, the traditional method is no longer able to calibrate the camera network, while the proposed one successfully estimates the poses of all the cameras in different rooms. In *Arrangement2*, the average calibration error of the proposed method is 5.2 cm, and it further improves to 4.1 cm after the depth image-based refinement.

In this experiment, the online calibration ran at real-time, and the offline refinement converged in about five seconds.

C. *Person tracking over non-overlapping camera views*

To demonstrate that the proposed method allows the integration of static and dynamic camera images in a unified frame, we conducted a person tracking experiment. Fig. 9 (a) shows the experimental environment, in which we placed three separated RGB-D cameras. Here, a subject walks over separated camera views while an observer holding a mobile camera follows the subject and captures them with the mobile camera. The static cameras track the subject as long as they are in their views, while the mobile camera complementary tracks them once they move out from the static camera views. By integrating the tracking results of the static and dynamic cameras, we can keep tracking the identity of the subject over the entire large room with a limited number of static cameras.

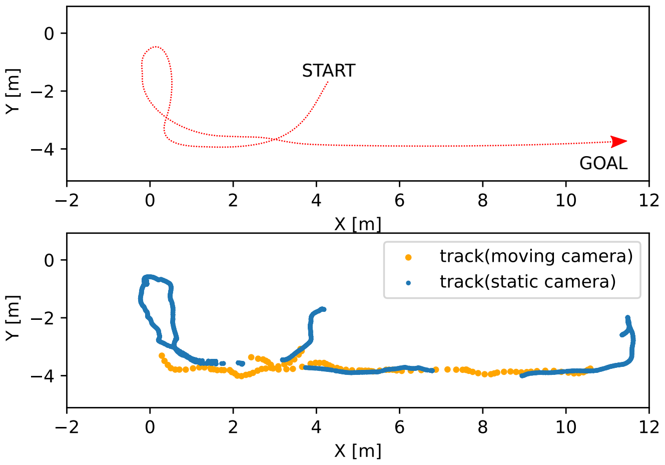


Fig. 11: The tracked person trajectory.

To track the mobile camera pose, we use the pose graph constructed for camera network calibration. We fix all the static camera and tag vertices in the pose graph and then add new dynamic camera vertices to the graph in the same way as the online calibration step. By performing graph optimization, the new dynamic camera vertices are aligned with respect to the static cameras, allowing us to obtain the current mobile camera pose from the optimization result.

We use the point cloud-based people tracking framework [1] to track the subject with static cameras. To track with dynamic camera, we first run *OpenPose* [26], a deep convolutional neural network-based person skeleton detection framework, and then project the detected ankle position onto the floor plane ($Z = 0$) using the estimated camera pose (see Fig. 10). To obtain a smooth trajectory, we apply a moving average filter to the tracked trajectory. Note that the skeleton detection process was performed offline because it requires more computational power than that for the laptop PC used in this experiment. However, the mobile camera pose estimation itself was running at real-time.

Fig. 11 shows the person trajectories tracked by the static and mobile cameras. When in the static camera views, the subject was tracked by the static cameras correctly. Even when they moved out from the static camera views, the system could keep tracking them with the mobile camera. We can see that the trajectories obtained by the static and mobile cameras are well aligned. This implies that the mobile camera frame is accurately aligned with respect to the camera network coordinates.

V. Conclusions

We proposed a calibration method for a camera network consisting of non-overlapping static cameras and a dynamic camera. In the proposed method, the dynamic camera, in which ego-motion is tracked by visual odometry, bridges non-overlapping cameras by observing fiducial tags seen by the static cameras. The depth image-based ICP and floor plane constraints are introduced to compensate for visual odometry errors. Through evaluations, it was confirmed that the proposed method shows a good calibration accuracy in both simulated and real environments.

We plan to apply the proposed method to a camera mounted on a mobile robot to keep tracking the robot position with respect to the camera network. It allows us to build a new human-robot interaction system based on the collaboration of camera network and mobile robotic systems.

Acknowledgment

This work was partially supported by a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] M. Munaro, F. Basso, and E. Menegatti, "Openptrack: Open source multi-camera calibration and people tracking for rgb-d camera networks," *Robotics and Autonomous Systems*, vol. 75, pp. 525–538, 2016.

[2] M. Carraro, M. Munaro, and E. Menegatti, "Skeleton estimation and tracking by means of depth data fusion from depth camera networks," *Robotics and Autonomous Systems*, vol. 110, pp. 151–159, 2018.

[3] K. Koide, E. Menegatti, M. Carraro, M. Munaro, and J. Miura, "People tracking and re-identification by face recognition for rgb-d camera networks," in *European Conference on Mobile Robots*. IEEE, 2017, pp. 1–7.

[4] M. Warren, D. McKinnon, and B. Upcroft, "Online calibration of stereo rigs for long-term autonomy," in *IEEE International Conference on Robotics and Automation*. IEEE, 2013.

[5] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009.

[6] R. Xia, M. Hu, J. Zhao, S. Chen, Y. Chen, and S. Fu, "Global calibration of non-overlapping cameras: State of the art," *Optik*, vol. 158, pp. 951–961, 2018.

[7] D. Makris, T. Ellis, and J. Black, "Bridging the gaps between cameras," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2004.

[8] A. Rahimi, B. Dunagan, and T. Darrell, "Simultaneous calibration and tracking with a network of non-overlapping sensors," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2004.

[9] N. Anjum, "Camera localization in distributed networks using trajectory estimation," *Journal of Electrical and Computer Engineering*, vol. 2011, pp. 1–13, 2011.

[10] H. Huang, N. Li, H. Guo, Y.-L. Chen, and X. Wu, "Calibration of non-overlapping cameras based on a mobile robot," in *International Conference on Information Science and Technology*. IEEE, 2015.

[11] F. Zhao, T. Tamaki, T. Kurita, B. Raytchev, and K. Kaneda, "Marker based simple non-overlapping camera calibration," in *IEEE International Conference on Image Processing*. IEEE, 2016.

[12] E. Ataer-Cansizoglu, Y. Taguchi, S. Ramalingam, and Y. Miki, "Calibration of non-overlapping cameras using an external SLAM system," in *IEEE International Conference on 3D Vision*. IEEE, 2014.

[13] T. Pollok and E. Monari, "A visual SLAM-based approach for calibration of distributed camera networks," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*. IEEE, 2016.

[14] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2016.

[15] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[16] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G: A general framework for graph optimization," in *IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.

[17] P. J. Huber, "Robust estimation of a location parameter," in *Springer Series in Statistics*. Springer, 1992, pp. 492–518.

[18] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[19] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback," *International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[20] K. Koide, J. Miura, and E. Menegatti, "A portable 3d lidar-based system for long-term and wide-area people behavior measurement," *International Journal of Advanced Robotic Systems*, 2019.

[21] 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 ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[22] L. Ma, C. Kerl, J. Stuckler, and D. Cremers, "CPA-SLAM: Consistent plane-model alignment for direct RGB-d SLAM," in *IEEE International Conference on Robotics and Automation*. IEEE, 2016.

[23] K.-L. Low, "Linear least-squares optimization for point-to-plane icp surface registration," *Chapel Hill, University of North Carolina*, vol. 4, no. 10, 2004.

[24] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2004.

[25] A. Rasouli and J. K. Tsotsos, "The effect of color space selection on detectability and discriminability of colored objects," *arXiv preprint arXiv:1702.05421*, 2017.

[26] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.