SELF-CALIBRATION IN VISUAL SENSOR NETWORKS EQUIPPED WITH RGB-D CAMERAS

Xiaoqin Wang, Y. Ahmet Şekercioğlu, Tom Drummond

ARC Centre of Excellence for Robotic Vision, Monash University, Australia

ABSTRACT

We consider the self-calibration problem (estimation of location and orientation of multiple camera sensors), in visual sensor networks equipped with RGB-D cameras. We propose two algorithms based on feature matching and relative pose estimation. First one uses Floyd-Warshall algorithm, and can accurately estimate the camera locations. On the other hand, the second algorithm is more scalable than the first one, and can be used in large networks if high accuracy is not an issue. Numerical results demonstrate that both algorithms, depending on the network topology, can be used for self-calibration in RGB-D equipped visual sensor networks.

Index Terms— Self-calibration, visual sensor network, RGB-D camera.

1. INTRODUCTION

Estimating the geometry of a visual sensor network (VSN) from the captured image information only, i.e. self-calibration [1, 2], is the prerequisite for many applications such as surveillance and object tracking. Different from the conventional VSNs, VSNs equipped with RGB-D sensors (e.g., Microsoft Kinect [3]), which enable capturing the color images with per-pixel depth information, provide fully 3D representation of the environments and thereby allow enhanced performance in conventional services and promise a wider range of the innovative applications. Self-calibration for VSNs with conventional cameras is a well developed area. However, to the best of our knowledge, no research work on self-calibration method for multiple RGB-D sensors has been done. In this paper, we propose two self-calibration algorithms for VSNs equipped with RGB-D cameras.

We consider scenarios where battery-powered mobile RGB-D sensors [4] are deployed to monitor and map a target region (an example is shown in Fig. 1). The interference between sensors can be canceled using a "shake'n'sense" approach [5]. Each sensor is able to communicate with each other in ad-hoc manner. A central node with high performance processor is also implemented in the network which can operate computationally expensive algorithms. Due to the limited communication bandwidth and battery-power supply of VSNs, self-calibration in VSNs should take the energy and bandwidth consumptions into consideration. Thus, we intend to achieve self-calibration while minimize the amount of transmitted information. The proposed self-calibration methods consist of the following steps: (1) extract color features at each sensor locally and send the descriptors of these features to the central node, (2) perform feature matching and generate a *Feature Matching Matrix* (FMM), (3) detect neighboring sensors based on FMM, (4) estimate the relative poses between neighbors and use selected relative poses to calibrate the overall system. We formulate the selection of relative poses as a shortest path problem, which consists of finding shortest path from a vertex to the other vertices in a edge-weighted graph that represents the sensors in the network. In contrast to conventional self-calibration algorithms for color cameras, our proposed algorithms determine sensors' locations and orientations in real world scale directly, and so they do not suffer from the scale ambiguity problem.

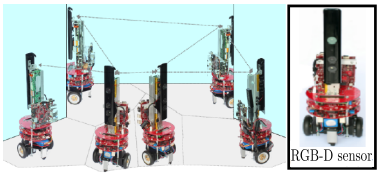


Fig. 1: A network of RGB-D sensors deployed to monitor and map a scene in 3D.

2. RELATIVE POSE ESTIMATION (RPE)

The location and orientation of one sensor in the other sensor's coordinate system is called relative pose. The relative pose of the RGB-D sensor b with respect to sensor a can be represented by a transformation matrix, $\mathbf{M}_{ab}$, in SE(3),

$$\mathbf{M}_{ab} = \begin{bmatrix} & \mathbf{R} & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

where $\mathbf{R}$ is a 3×3 rotation matrix and $\mathbf{t}$ is a 3×1 translation vector.

There are various methods to estimate the relative pose between two imaging sensors. In this paper, we adopt the

This work was supported by the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016).

Relative Pose Estimation (RPE) algorithm presented in one of our earlier works [4]. This algorithm is specially designed for RGB-D sensors and has robust performance with minimized bandwidth consumption. It is an improved variant of *Iterative Closest Points* (ICP) which operates between two RGB-D sensors with overlapping FoVs. This distributed, peer-to-peer algorithm determines the relative pose through explicit registration of surface geometries extracted from the depth images captured by two sensors. More details about this approach can be found in [4].

3. SELF-CALIBRATION ALGORITHMS

The RPE algorithm can be extended to operate in the circumstance with more than two sensors. For example, sensors a - b, and b - c are two pairs of sensors with overlapping FoVs. Sensor b's pose in sensor a's coordinate system can be derived as $\mathbf{M}_{ab}$. Sensor c's pose in sensor b's coordinate system is $\mathbf{M}_{bc}$. Then, sensor a can determine sensor c's pose as

$$\mathbf{M}_{ac} = \mathbf{M}_{bc}\mathbf{M}_{ab}. \tag{2}$$

In this example, sensor b is the *intermediate node* to determine the relative pose between sensors a and b.

In order to accomplish self-calibration based on this approach, we first need to determine the sensors with overlapping FoVs. Then, sensors are grouped in pairs to determine the relative pose. At last, each sensor can obtain the geometric topology of the whole networks.

3.1. Neighbor Detection

We define the sensors with overlapping FoVs as neighbors. One sensor's neighbors can be detected through comparing observed color images based on matching features. There are three steps involved for this process: feature detection, feature description, and feature matching. The first two steps are operated by each sensor locally. Taking both processing speed and accuracy into account, we implement FAST [7] for feature detection and ORB [8] for feature description on every sensor. Then, instead of transmitting the complete images, each sensor sends the feature descriptors to the central node to minimize the transmission load.

The central node performs feature matching between every two sets of the feature descriptors. In order to match image features more reliably and get rid of the mismatched outliers, we adopt the symmetrical matching scheme and the epipolar constraint between two views. In the symmetrical matching scheme, the correspondences between two sets of feature descriptors are established bidirectionally. One group of correspondences is generated from matching the first feature set to the second feature set. The other group is produced from matching the second feature set to the first feature set. For a pair of matched features to be accepted, two features must be the best matching candidate of each other in both directions. Then, we estimate the fundamental matrix associated with the images by using the survived matches and RANSAC [9] strategy. Ultimately, only the matches which agree with the fundamental matrix are kept as the good matches. An intuitive example of feature matching with and without refinement process is shown in Fig. 2.

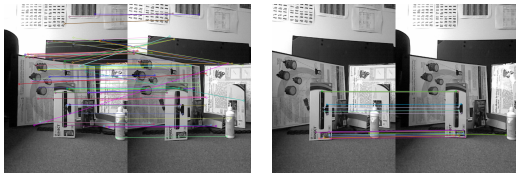
Table 1: FMM and NM of a network with three sensors.

FMM

No.	1	2	3
1	×	n_{12}	n_{13}
2	n_{21}	×	n_{23}
3	n_{31}	n_{32}	×

NM

No.	1	2	3
1	×	w_{12}	w_{13}
2	w_{21}	×	w_{23}
3	w_{31}	w_{32}	×



(a) **(b)**

Fig. 2: Feature matching results: (a) without refinement, (b) with refinement.

After conducting the above process on each two sets of feature descriptors, a FMM can be constructed. As shown in Table 1, FMM is symmetric with each element (n_{ij}) representing the number of matched features between the images captured by sensors i and j. The diagonal elements represent the feature matching with itself, thus they are negligible. Let $n_{\max}$ represent the largest number of matched features in FMM. By assuming the texture features are normally distributed in the scene, the larger $n_{ij}/n_{\max}$ indicates better overlapping in two sensors' FoVs. Based on FMM and criteria in Eq. 3, the Neighbor Matrix (NM) can be generated. Smaller w_{ij} indicates smaller uncertainty value in relative pose estimation between two neighboring sensors.

$$w_{ij} = \begin{cases} 1 & \text{if } n_{ij}/n_{\max} \geq 0.8 \\ 1.5 & \text{if } 0.8 > n_{ij}/n_{\max} \geq 0.7 \\ 2.4 & \text{if } 0.7 > n_{ij}/n_{\max} \geq 0.6 \\ \infty & \text{if } 0.6 > n_{ij}/n_{\max} \end{cases} \tag{3}$$

3.2. Selection of Relative Poses

The RPE algorithm only introduces a small amount of uncertainties and communication overhead when it is operated between two sensors. However, if two sensors require too many intermediate nodes to determine each other's poses, the accumulation error will significantly influent the accuracy of the result. In order to ensure each sensor has the reliable knowledge of the other sensors' locations and orientations, we require to select the relative poses which introduce the smallest overall amount of uncertainties to calibration the system.

This problem is transformed to the all-pairs shortest path problem. According to the NM, we can generate a sensor dependency graph, $G = (V, A)$, with sensors as the vertices.

There is an edge between any two sensors iff they are neighbors. The weight of the edge linking sensor i and j is w_{ij}, which indicates the uncertainty level in RPE. As NM is symmetrical, all edges is treated as bidirectional.

3.2.1. Method 1

We propose a method based on Floyd-Warshall algorithm to determine the shortest path between every two vertices. In the propose approach, we first generate $Dist$ as a $|V| \times |V|$ array of minimum distance and initialize $Dist$ according to NM. Then, we adopt Floyd–Warshall algorithm to determine the shortest paths between every pair of sensors and update $Dist$. However, in some circumstances, due to limited texture features in the captured images, some sensors can be isolated which cannot find its links to the other sensors. If more than a certain threshold, γ, of the sensors are isolated, we will lower the thresholds in Eq. 3 and operate Floyd–Warshall again. Otherwise, we will link each isolated sensor to its *principal neighbor* which has the largest number of matched features with this isolated sensor and finds the shortest paths to the other sensors, and then update $Dist$ array. γ is influenced by the environment and sensors' poses. We set γ at 30% for general cases. In some scenarios, some sensors do not have enough overlapping area with any other sensors, then we leave these sensors as isolated sensors. These isolated sensors cannot be self-calibrated. At last, the central node will send the information of linked sensor pairs to each sensor, and sensors will operate relative pose estimation algorithm according to the requests. The following summarizes the pseudo code of the algorithm.

Algorithm 1 Method based on Floyd-Warshall Algorithm

I. Initialization Phase

1: Generate the sensor dependency graph, $G = (V, A)$, based on NM. Let $Dist$ be a $|V| \times |V|$ array of minimum distance initialized according to NM.

II. Decision Phase

```
1: while less than 1 − γ sensors are linked do
2:     operate Floyd-Warshall Algorithm
3:     if more than γ of sensors are isolated then
4:         lower the threshold in Eq. 3,
           update NM and initialize Dist accordingly.
5:     else
6:         link each isolated sensor to its principle neighbor
           and update Dist.
7:     end if
8: end while
```

Though this method can detect the best path between every two sensors, its disadvantage is obvious– due to the cycles may exist in the graph, the relative pose estimation algorithm need to run $(|V| - 1)^2/2$ times for the worst case in which each sensor estimates the relative poses of the others. So the time complexity of the complete self-calibration scheme is $O(V^2)$.

3.2.2. Method 2

We now present another method which links sensors to form a hierarchical tree structure. Different from method 1, this method first selects a *primary sensor* as the root and then find the shortest paths from the root to the other sensors. Thus, each sensor can obtain the other sensors' poses information through searching according to this shortest-path tree.

Determining the primary sensor is a two-step process. First, the sensors with the largest number of neighbors are picked. Then, the sensor with the smallest average weight among the picked sensors is set as the primary sensor. After the primary sensor is selected, the shortest paths between the primary sensor and the other sensors can be determined by the central node. This single-source shortest path problem can be solved by many approaches [10, 11]. In our method, we adopt the A* search algorithm [12]. The remaining process is similar to method 1. The following summarizes the pseudo code of the algorithm. The central node will send the information of the established shortest-path tree to each sensor. Then, sensors will operate RPE algorithm according to the shortest-path tree. A simple example of this working process is shown below. Fig. 3 depicts a shortest-path tree of a network. Sensors operate RPE algorithm to derive the relative poses $\mathbf{M}_{ab}$, $\mathbf{M}_{ac}$, and $\mathbf{M}_{cd}$ according to the tree. By using these three pose matrices, the relative pose between every two sensors in the network can be determined. For instance, sensor d's location and orientation in sensor b's coordinate system can be derived as $\mathbf{M}_{bd} = \mathbf{M}_{ad}\mathbf{M}_{ba} = \mathbf{M}_{cd}\mathbf{M}_{ac}\mathbf{M}_{ab}^{-1}$.

Algorithm 2 Method based on shortest-path tree

I. Initialization Phase

1: Same as the initialization phase in method 1.

II. Decision Phase

```
1: while less than 1 − γ sensors are linked do
2:     select primary sensor, operate A* search Algorithm.
3:     if more than γ of sensors are isolated then
4:         lower the threshold in Eq. 3,
           update NM and initialize Dist accordingly.
5:     else
6:         link each isolated sensor to its principle neighbor
           and update Dist.
7:     end if
8: end while
```

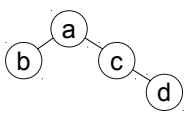
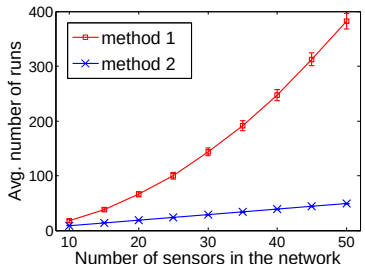
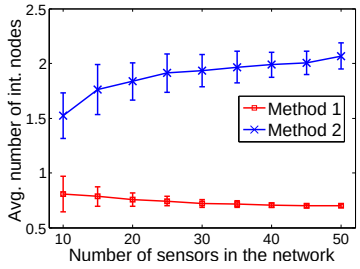


Fig. 3: Example of a shortest-path tree.

In this method, the RPE algorithm only requires to perform $|V| - 1$ runs for the worst case. Thus, the time complexity of the complete self-calibration scheme is $O(V)$. Method 2 requires to operate RPE algorithm for a much fewer overall number of runs than method 1. Therefore, method 2 incurs lower communication overhead than method 1.



(a) Average number of runs that RPE algorithm processes



(b) Average number of intermediate nodes as a function of the number of sensors

Fig. 4: Simulation results

4. EXPERIMENTAL RESULTS

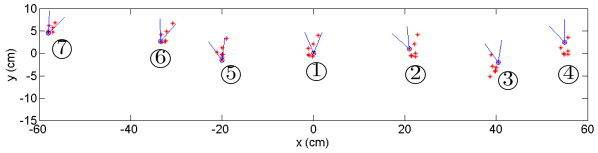
We conducted both simulations and real world experiments to evaluate performance of the proposed methods. In simulations, we compared the time complexity and the average numbers of intermediate nodes required by two methods. In experiments, we focused on analyzing the accuracy of two methods.

We simulated different networks with increasing number of sensors from 10 to 50. The sensors have random overlapping FoVs. Two proposed methods were applied to realize self-calibration of the networks. We evaluated the performances of two methods through comparing the numbers of intermediate nodes and the numbers of runs that the RPE algorithm operates. The results presented in Fig. 4 are averaged over 100 runs of the simulations with vertical bars indicating the variance.

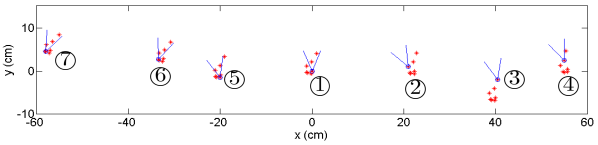
According to Fig. 4a, we can see that the average number of RPE algorithm runs required by method 1 increases more significantly than method 2. It is because that each sensor has more neighbors when the number of sensors increases. Method 1 requires to estimate the relative pose between nearly every two neighbors to realize the optimal shortest paths. While method 2 only need to operate RPE algorithm for $|V| - 1$ runs. As shown in Fig. 4b, when the number of sensors increases, the average number of intermediate nodes required by method 1 decreases, while this number of method 2 increases. Because when the number of sensors raises and the scene becomes crowded, sensors have more neighbors and can estimate their relative poses directly without intermediate node in method 1. However, the shortest path tree constructed in method 2 becomes larger when the number of sensors increases. Two neighboring sensors, which can be the leaves with different parents, still have to use the root as the intermediate node to determine the relative pose. So the average number of intermediate nodes in method 2 increases.

In the experiment, we used the color and depth images captured by an experimental VSN platform consisting of 7 RGB-D sensors. This platform is developed in Monash University's Wireless Sensor and Robot Networks Laboratory (WSRNLab) [4]. The color images captured by different sensors are illustrated in Fig. 5a. The results of two methods are presented in Fig. 5b and c. The groundtruth locations and orientations are depicted as blue circles with line segments. The estimated locations are shown as red crosses. The estimated orientations are not shown in Fig. 5, as they make the graphs messy and unclear. RMS errors of locations and orientations estimated by method 1 are 4.19cm and $4.57°$. RMS errors of locations and orientations estimated by method 2 are 5.20cm and $6.72°$. RPE is performed 9 runs by method 1, and it is operated 6 runs by method 2. We can see that the method 1 is more accurate than method 2. However, it requires to operate RPE for a larger number of times and incurs higher communication overhead.

(a) images observed by different sensors



(b) method 1



(c) method 2

Fig. 5: Real world experimental results.

5. DISCUSSION AND CONCLUSION

This paper is the first work which addresses the self-calibration problem in VSNs consisting of RGB-D cameras. Two methods are proposed for networks with various requirements and constraints. Though the second method cannot be expected to achieve as good a performance (in terms of accuracy) as the first method, it is more computationally scalable and does not incur high communication overhead as required by the first method. Application areas in which the sensor node pose information can be used include 3D reconstruction, image-based modeling, and multi-view object tracking.

6. REFERENCES

[1] Shafique, K. and Hakeem, A and Javed, O. and Haering, N., "Self Calibrating Visual Sensor Networks", *IEEE Workshop on Applications of Computer Vision*, 2008, pp. 1,6.

[2] Kelly, J and S Sukhatme, G., "Visual-Inertial Sensor Fusion: Localization, Mapping and Sensor-to-Sensor Self-calibration", *Intl. J. Robotics Research*, 2011, **30**, pp. 56-79.

[3] Han, J. and Shao, L and Xu, D. and Shotton, J., "Enhanced Computer Vision With Microsoft Kinect Sensor: A Review", *IEEE Trans. Cybernetics*, 2013, **43**, pp. 1318-1334.

[4] "Wireless Sensor and Robot Networks Laboratory (WSRNLab)", http://wsrnlab.ecse.monash.edu.au.

[5] Butler, D. A., Izadi, S., Hilliges, O., Molyneaux, D., Hodges, S., and Kim D. "Shake'n'sense: reducing interference for overlapping structured light depth cameras", *ACM Conf. Human Factors in Computing Systems*, 2012, pp. 19331936.

[6] X. Wang and Y. A. Şekercioğlu and T. Drummond, "A Real-Time Distributed Relative Pose Estimation Algorithm for RGB-D Camera Equipped Visual Sensor Networks", *ACM/IEEE Intl. Conf. Distributed Smart Cameras*, 2013.

[7] Rosten, E. and Drummond, T., "Machine Learning for High-Speed Corner Detection", *Computer Vision-ECCV*, 2006.

[8] Rublee, E., Rabaud, V., Konolige, K., Bradski, G.,"ORB: An Efficient Alternative to SIFT or SURF", *IEEE Intl. Conf. Computer Vision*, 2011, pp. 2564-2571.

[9] Fischler, M. and Bolles, R., "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Comm. of the ACM*, 1981, pp. 381-395.

[10] Dijkstra, E.,"A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik*, 1959, pp. 269-271.

[11] Dijkstra, E.,"On a Routing Problem", *Quarterly of Applied Mathematics*, 1958, pp. 87-90.

[12] Hart, P. E., Nilsson, N. J., Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Trans. Systems Science and Cybernetics*, 1968, pp. 100-107.