Autonomous, Lightweight Calibration of Visual Sensor Networks with Dense Coverage

Jennifer Simonjan, Bernhard Rinner
Alpen-Adria-Universität Klagenfurt, Austria
Email: {firstname.lastname@aau.at}
Klagenfurt, Austria

***Abstract*—We present an algorithm for autonomous network calibration of visual sensor networks, which become more and more pervasive since they can be found in various everyday life environments. The proposed algorithm works in a fully decentralized way and minimizes usage of cost-intensive vision algorithms. To achieve network calibration, our approach relies on jointly detected objects and geometric relations between camera nodes. Distances and angles are the only information required to be exchanged between nodes. The process works iteratively until cameras have determined the relative position and orientation of their neighbors. Preliminary results are demonstrated using our visual sensor network simulator.**

I. Introduction

Visual sensor networks (VSNs) consist of spatially distributed smart camera devices, which are capable of capturing and processing images of a scene. A network of multiple cameras provides a variety of information from several viewpoints. VSNs are becoming ubiquitous and pervasive since they can be found in many applications including home automation, ambient assisted living, surveillance and entertainment systems. The camera nodes themselves are getting smaller, more powerful and cheaper nowadays, enabling a new generation of large-scale and efficient applications. In the deployment phase of a VSN, the network calibration is of important interest in order to enable a quick and simple installation. Network calibration is concerned with establishing relations between neighboring nodes, in order to gather knowledge about the network topology. This process should be done autonomously to enable ad-hoc deployment in dynamic environments.

In our work we target autonomous network calibration of highly scalable VSNs with dense coverage [1]. Our network calibration technique enables cameras to learn the relative position and orientation of neighboring nodes, without the need for a-priori knowledge or user-interaction. Thereby, we define neighbors as nodes with overlapping fields of view (FOVs). The approach works without complex vision algorithms and keeps communication costs low in order to save resources. Further, the algorithm is fully decentralized and based on simple distance/angle estimations, achieving several advantages over other approaches: First, there is no user-interaction required. Second, networks become highly scalable and third, the resource efficiency is increased significantly since the algorithm operates solely in the local neighborhood avoiding cost-intensive processing and communication in large-scale networks.

Autonomously calibrating decentralized VSNs has recently gained interest in the research community since calibrated networks enable higher robustness and smarter applications. Detmold et al. [2] presented an approach which is capable of estimating the activity topology for a network of thousands of cameras. Each camera learns which cameras are within the neighborhood but no camera positions and orientations are estimated. The approach is centralized which limits the scalability and increases the required installation effort of the network noticeably. Esterle et al. [3] proposed a decentralized solution for learning vision graphs of VSNs. Deravajan et al. [4] proposed a framework for intrinsic (focal length) and extrinsic (3D location and orientation) calibration of cameras, which is based on feature extraction and matching. Liu et al. [5] estimate the transformation between two cameras using noisy foreground blobs and relying on a joint optimization framework with robust statistics. Borra and Fagani [6] calibrate camera networks in a distributed fashion without the need for complex vision algorithms. However, the cameras are calibrated with respect to a global reference frame. Anjum and Cavallaro [7] presented an algorithm for external calibration of camera networks with non-overlapping FOVs by estimating trajectories in unobserved regions.

The rest of the paper is organized as follows. In section II, we present our approach for position and orientation estimation of neighboring cameras. Section III shows and discusses preliminary simulation results. We outline future work in section IV and conclude in section V.

II. Approach

Overview

The goal of our network calibration approach is to establish a common coordinate system which contains positions and orientations of all cameras within the network. For that purpose, cameras operate on local coordinate systems while estimating relative positions and orientations of neighbors. The nodes solely rely on local interactions with their neighbors, simple vision based detections and geometric relations. There is no a-priori knowledge about node positions/orientations and no global reference available.

Calibration algorithm

We divide the calibration process into two levels: object-based estimation and network-wide calibration. Object-based

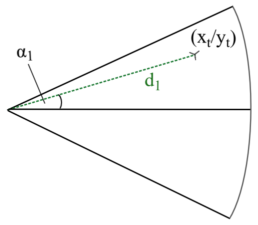


Fig. 1. Distance d_1 and angle α_1 of camera c_1 to a detected object.

estimation is done whenever a camera detects an object in its own FOV. Network-wide calibration includes communication and is done to relate to neighboring nodes and to spread calibration information. Throughout the whole process, each camera operates on a local coordinate system with its own position in the origin and an orientation of 0°. Relative positions and orientations of neighboring cameras (those which have overlapping FOVs) are determined, added to the local coordinate system and spread through the network.

Assumptions: All cameras are placed at the same altitude and position estimation is done in 2D (will be extended in the simulator in future work). The camera network is fully distributed and synchronized and the cameras have communication capabilities such as WiFi. We assume a single object of known size (will be relaxed as a next step) moving through the network and a simple vision based detection running on each camera. Cameras are able to re-identify objects.

Object-based estimation: When an object is detected, cameras estimate the distance d and the angle α, as shown in Figure 1, to the object. Since each camera assumes itself at the origin with orientation 0°, α is positive if the object was detected in the right part of the FOV and negative otherwise.

The size of the object is required to determine the distance d. Currently, we assume that object sizes are known. However, we will extend our approach by estimating the object size based on statistical calculations. Knowing the size, a camera can measure the vertical angle and therefore calculate the distance d. The object position (x_t/y_t) is clearly defined by radius d and angle α. When a camera has finished the estimation, it broadcasts its distance d and horizontal angle α, which will trigger the calibration process. The pseudo code of the object-based estimation is shown in Algorithm 1 and runs on each camera in the network.

Algorithm 1 Object-based estimation algorithm

```
if object is detected then
    generate time stamp $t_i$
    generate object identifier $o_i$
    estimate distance $d_i$ to object
    estimate horizontal angle $\alpha_i$ to object
    broadcast message including $t_i$, $o_i$, $d_i$ and $\alpha_i$
end if
```

Network-wide calibration: In the network-wide calibration phase, nodes send and receive object information and use it to estimate relative positions and orientations of neighbors. To ease the explanation, we assume two cameras c_1 and c_2 with overlapping FOVs. Both cameras have seen the object at the same time and completed the steps of Algorithm 1.

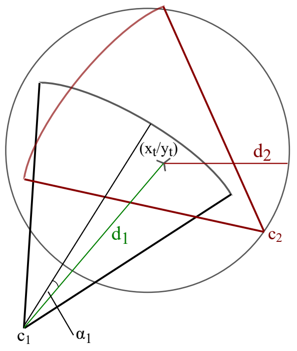


Fig. 2. Circle of camera c_1 to estimate the position of camera c_2. Camera c_2 lies somewhere on this circle.

Determining positions: Camera c_1 thus now receives the distance d_2 of camera c_2 and starts with the position estimation. Camera c_1 estimates the position of camera c_2 somewhere on the circle with radius d_2 and center (x_t/y_t) (defined by d_1 and α_1). The estimation is done for every joint object detection (i.e., multiple objects, or one object at different time instances). Figure 2 shows the estimated circle of camera c_1 for the common object detection to determine the position of camera c_2. Camera c_2 lies somewhere on this circle.

Every time the cameras jointly detect an object, the calculations are repeated, resulting in further circles. The intersections of all circles are calculated. The algorithm works iteratively until there is just one intersection left, which intersects with every single circle - this is the relative position of the neighboring camera. Summarized, each camera ends up with one circle per common object detection. After multiple detections, the intersection of all circles leads to the position of the neighboring camera. Each camera does these calculations for all neighbors, resulting in a decentralized calibration approach.

Determining orientations: Cameras use received angles and object positions to determine the orientation of their neighbors. Again, lets assume cameras c_1 and c_2 have an overlapping FOV and detected a common object. The object shown in Figure 3 is in the left part of the FOV of camera c_2 resulting in a negative angle α_2 (i.e., $-20°$).

Camera c_1 receives besides the distance d_2 also the angle α_2 of camera c_2. If camera c_1 has finished determining (x_2/y_2), the calculation of the angle β, as shown in Figure 3, is simple. Knowing the neighbor's position (x_2/y_2) and the position of the object (x_t/y_t), camera c_1 calculates the angle between those two points w.r.t. its local coordinate system and adds the received angle α_2 in order to determine β, which is the difference in orientation. To enable large-scale calibration, locally determined position/orientations are spread trough the network. The pseudo code of the network-wide calibration is shown in Algorithm 2 and runs, as Algorithm 1, on each camera in the network.

The computational cost for the calibration is one message

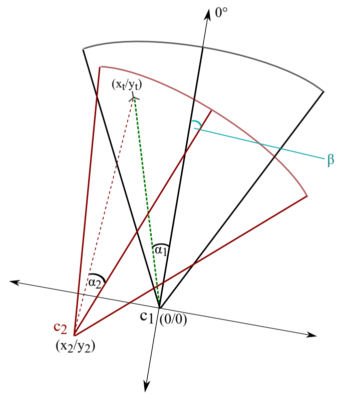


Fig. 3. Local coordinate system of camera c_1 with known positions (x_2/y_2) and (x_t/y_t). The angle β depicts the difference in orientation, in which we are interested.

Algorithm 2 Network-wide calibration algorithm

```
On object description message is received
  if if object $o_i$ has also been locally detected at $t_i$ then
    if position of sender node is unknown then
      estimate position of sender node
    else
      estimate orientation of sender node
    end if
    broadcast calibration information
  end if
```

per object detection per neighbor, one broadcast message per finished position/orientation estimation and geometrical calculations including sine and cosine.

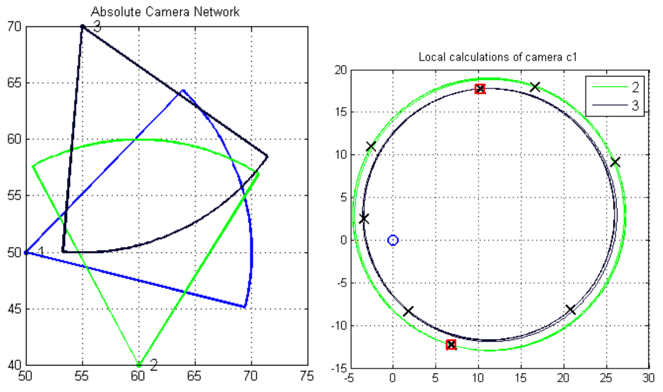
III. Preliminary results

We developed a VSN simulator to enable easy and fast development and testing of algorithms. First tests were done using this simulator. Comparative evaluations of a real camera network will follow. The simulator can simulate a network of static cameras with arbitrary FOVs and certain internal parameters such as resolution and sensor range. The basic camera model was adopted from Dieber et al. [8]. However, we additionally model the vertical angle of cameras and the vertical size of objects. Further, the simulator allows to add moving objects to the scenario. Those objects can either move randomly on the ground plane or follow specific predefined way-points. So far, objects are modeled as 1x1 boxes and the detection location of an object is defined by the lower middle point of the bounding box.

Simulation scenario: Due to page limits, we will only show one of our tested scenarios. In this scenario we simulated three cameras with overlapping FOVs and arbitrary orientations. One object was moving in the area of interest following specific way-points. The way-points were constructed from the PETS 2009 dataset[1] to ensure a real world movement pattern.

[1]http://www.cvg.reading.ac.uk/PETS2009/a.html

Simulation results: To enable a meaningful evaluation and comparison, we plotted results from local camera calculations as well as ground truth data from the simulator. Figure 4 (a) shows the input scenario depicting the FOVs of the three cameras in different colors and their positions. Figure 4 (b) shows the position estimations of camera c_1 for this scenario. Since those calculations take place locally on the camera, the figure shows the local coordinate system of camera c_1. The small circle depicts the position of camera c_1 itself. The large circles depict the position estimations for neighboring cameras c_2 and c_3, whereby the inner black circles belong to camera c_3 and the green ones to camera c_2. Calculations for multiple neighbors are done independently - however, we plotted them into one figure. The crosses show all intersections where at least two of the circles intersect with each other. The red squares depict the intersections which intersect every single circle (per neighbor). Thus, the squares determine the estimated position of the neighbors.

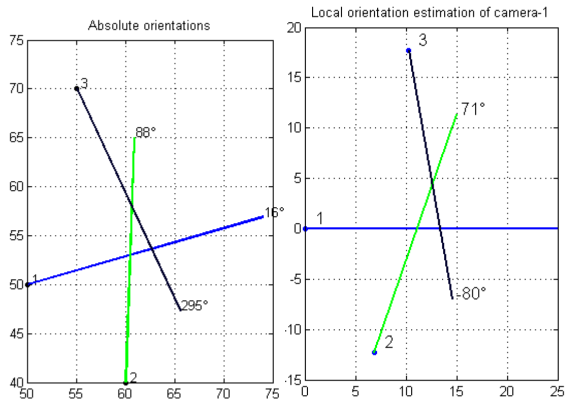


(a) (b)

Fig. 4. (a) The ground truth data. (b) Local measurements of camera c_1. The small circle shows the position of camera c_1. The large circles are the calculations for neighboring cameras c_2 and c_3. The crosses on the circles depict the intersections, whereby the red squares depict the determined neighbors positions.

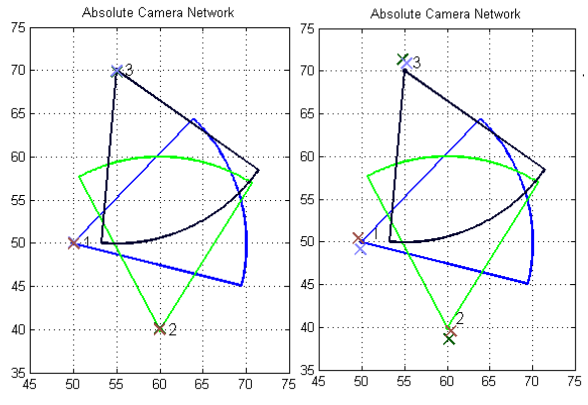
For evaluations, we calculate the distances between the cameras. Figure 4 (a) shows that camera c_1 is located at $(50/50)$ and camera c_2 at $(60/40)$. Calculating the distance between those two points results in a distance of 14.14. In Figure 4 (b) camera c_1 is located at the origin and camera c_2 at $(6.8/-12.3)$. In that case the distance is given as $\sqrt{6.8^2+(-12.3)^2} = 14.05$. For scenarios without any error injection, the resulting distances match (except to rounding errors). Thus, we can show that our algorithm can reconstruct the real topology using only local knowledge.

Additionally to distances, also the orientation is of interest. After the position of a neighbor has been determined, the orientation is estimated. Again, we compare local results with the ground truth. Figure 5 (a) shows the absolute orientations of all cameras. Figure 5 (b) shows the local orientation estimations of camera c_1. The difference in orientation for the cameras c_1 and c_2 is $88° - 16° = 72°$ in Figure 5 (a) and $71° - 0° = 71°$ in Figure 5 (b). Thus, the geometry of sensor



(a) (b)

Fig. 5. (a) The ground truth data. (b) Local orientation measurements of camera c_1. The ground line depicts camera c_1 with orientation 0°. The other lines show the relative orientations of the neighboring cameras.



(a) (b)

Fig. 6. Both figures show the FOVs of cameras c_1, c_2 and c_3. We plotted the locally estimated positions (crosses) of all cameras into the same figures. (a) Scenario without errors in the object detection. (b) We introduced an error in the distance estimation - which was a deviating object size estimation by 0.1m.

directions matches except to small rounding errors.

Now, we are interested in the robustness of the approach. In a real camera network, there will be various sources of errors, which we will try to model within the simulator in order to get an idea about robustness and real world behavior of the system. The results of our first robustness measurements are shown in Figure 6. Again, the camera network is depicted by three FOVs in different colors. However, we now added also colored crosses to the plot, which are the locally calculated positions of all cameras (transformed into the same coordinate system). Figure 6 (a) shows the scenario without any error. As we can see, the locally estimated positions (crosses) match the real positions and the average deviating distance is only 0.15m. This deviation is due to the distance estimation in Algorithm 1, which is an approximation, and due to rounding errors. Figure 6 (b) shows the results for an error of 0.1m in object size estimation. The crosses do not fit the right positions anymore and the deviating distance increased to 2m.

Discussion: The algorithm shows promising results while being highly resource efficient. It operates solely within the local neighborhood, affecting computational cost only if the number of neighbors is increased but not with increasing network size.

IV. Future work

Currently we are in the process of developing a solution that enables cameras to find neighbors in the network. This solution will rely on computer vision and learning techniques. Afterwards, we will develop a statistical based algorithm able to estimate object sizes, similar to what has been done by Liu et al. [5]. This will extend the object model to a realistic one. The object detections and the statistical estimations will enable cameras to differentiate between objects and thus allow multiple objects in the area of interest. We will test the approach by introducing various error sources, using different network sizes and numbers of objects. Further, we will deploy our algorithms to a camera network to test them under real conditions.

V. Conclusion

In this paper we presented an autonomous network calibration approach for distributed visual sensors networks. Cameras learn positions and orientations of neighboring cameras in a fully decentralized way. For that purpose they solely rely on geometric relations and basic object detections without the need for complex and cost-intense vision algorithms. We believe that the importance of user friendly and large-scale camera networks will increase in the near future since those networks become more and more part of the human's everyday life.

Acknowledgements

This work is supported by the research initiative 'Mobile Vision' with funding from the Austrian Federal Ministry of Science, Research and Economy and the Austrian Institute of Technology.

References

[1] J. Simonjan, "Towards large-scale pervasive smart camera networks," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, 2015, pp. 253–255.

[2] H. Detmold *et al.*, "Topology estimation for thousand-camera surveillance networks," in *Proceedings of the 1st ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, 2007, pp. 195–202.

[3] L. Esterle, P. R. Lewis, X. Yao, and B. Rinner, "Socio-economic vision graph generation and handover in distributed smart camera networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 2, p. 20, 2014.

[4] D. Devarajan, Z. Cheng, and R. J. Radke, "Calibrating distributed camera networks," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1625–1639, 2008.

[5] J. Liu, R. T. Collins, and Y. Liu, "Robust autocalibration for a surveillance camera network," in *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV)*, 2013, pp. 433–440.

[6] D. Borra and F. Fagnani, "Asynchronous distributed calibration of camera networks," in *Proceedings of the IEEE European Control Conference (ECC)*, 2013, pp. 754–759.

[7] N. Anjum and A. Cavallaro, "Automated localization of a camera network," *IEEE Intelligent Systems*, vol. 27, no. 5, pp. 10–18, 2012.

[8] B. Dieber, C. Micheloni, and B. Rinner, "Resource-aware coverage and task assignment in visual sensor networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 10, pp. 1424–1437, 2011.