Distributed Visual Sensor Network Calibration based on Joint Object Detections

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

***Abstract*—In this paper we present a distributed, autonomous network calibration algorithm, which enables visual sensor networks to gather knowledge about the network topology. A calibrated sensor network provides the basis for more robust applications, since nodes are aware of their spatial neighbors. In our approach, sensor nodes estimate relative positions and orientations of nodes with overlapping fields of view based on jointly detected objects and geometric relations. Distance and angle measurements are the only information required to be exchanged between nodes. The process works iteratively, first calibrating camera neighbors in a pairwise manner and then spreading the calibration information through the network. Further, each node operates within its local coordinate system avoiding the need for any global coordinates. While existing methods mostly exploit computer vision algorithms to relate nodes to each other based on their images, we solely rely on geometric constraints.**

I. Introduction

Visual sensor networks (VSNs) consist of spatially distributed and interconnected smart camera devices, which are not only capable of capturing images, but can also perform processing and communication. The camera nodes themselves are getting smaller and cheaper nowadays, enabling a new generation of large-scale applications such as home automation, environmental monitoring or entertainment systems. These applications found their way into our everyday life, requiring the networks to provide easy handling in terms of installation and calibration. Thereby, we consider network calibration as the process of gathering information about the topology, node positions and orientations. A calibrated network is the basis for many algorithms such as routing, data fusion and load balancing. In order to support dynamic networks, calibration techniques should also be scalable.

In this paper, we present a fully distributed, autonomous and efficient network calibration approach as well as evaluation results using our VSN simulator. Our technique enables cameras to learn relative positions and orientations of all other nodes in the network without relying on any global coordinate system or explicit user-interaction [1]. Most of the existing methods mainly rely on computer vision tasks, such as matching images of neighboring cameras, to calibrate cameras networks. We aimed for an approach based on simple geometric computations, avoiding complex computer vision algorithms and keeping communication and computational costs low.

Our algorithm works without any central entity and relies solely on simple distance/angle estimations. We can thus achieve main advantages over related approaches: i) no need for user-interaction, ii) simple and efficient geometric computations, iii) high scalability, iv) no need for a-priori network information or global coordinate systems and v) applicability to large-scale networks since processing is only done in the local neighborhood.

The remainder of the paper is organized as follows. Section II discusses related work and section III introduces the problem statement. Section IV presents our basic network calibration approach. The sensor error model and a method proposed to handle this error are discussed in section V. The evaluation results are presented in section VI. We conclude and outline future work in section VII.

II. Related work

Autonomous network calibration (also referred to as network localization) in sensor networks is of high interest in the research community, since calibrated networks enable higher robustness for applications [2], [3]. This is also true for visual sensor networks, which are often deployed in very large-scale environments or which even have to deal with dynamic network configurations due to mobile nodes. In general, two categories of network calibration can be distinguished based on whether neighborhood is represented by links or distance and orientation information.

i) network topology: estimation of the links between nodes
ii) network calibration: estimation of the positions and/or orientations of nodes

In case of network topology estimation, the topologies are often represented as graphs which do not contain position and orientation information [4]–[6]. Since our focus lies on calibration techniques which can estimate node positions and orientations, we present here only algorithms which are capable of such a detailed network calibration.

One calibration example is the approach of Lv et al. [7], in which cameras compute the vertical vanishing points and the horizon line from the motion of a walking human. This information is then used to compute intrinsic and extrinsic camera parameters.

Another example is the algorithm introduced by Liu et al. [8] for single cameras. It uses the height distribution of multiple humans in a moderately crowded scene to generate

a mapping between 3D objects and their projections in the 2D image plane. The authors developed a robust algorithm to estimate the pedestrians' height and orientation distribution, which was then used to intrinsically and extrinsically calibrate a camera. The approach was extended to camera networks in [9]. All cameras are calibrated to their own local coordinate system independently and then they register to a shared global world coordinate system.

The approach proposed by Possegger et al. [9] is able to extrinsically calibrate a network of static and pan-tilt-zoom (PTZ) cameras. For that purpose, the cameras rely on correspondences between the tracks of walking humans, from which they compute all head and foot locations. These locations are then used to estimate the extrinsic parameters of the cameras.

Ortega et al. [10] proposed a method for external calibration in outdoor camera networks with small or no overlapping fields of view (FOVs). The algorithm is based on matching the 3D lines of images, which were computed from dense 3D point clouds of the scene. However, the user has to manually obtain the nominal calibration on an aerial view in the first stage.

Yin et al. [11] introduced a semi-automatic calibration approach, which combines tracked blobs with user-selected features to recover camera homographies. The scene information is then used to achieve object correspondences across multiple views. Recovering homographies is an even more detailed calibration approach than estimating positions and orientations.

Guan et al. [12] proposed a VSN calibration method, which analyzes the tracks of pedestrians. Using these tracks, cameras first estimate the 3D positions of pedestrians in their local coordinate system. Next, the local coordinate systems are transformed to each other in a pairwise fashion. A main advantage is, that this method can handle the case where persons are moving along a straight line (usually introducing the co-planarity problem). The general idea of this work is similar to ours, however their main focus lies on the computer vision part and ours on networking and efficiency.

Most camera network calibration techniques use cost-intensive computer vision methods such as 3D reconstruction, work in a centralized manner or rely on user-input or global coordinate systems. Our main interest is to develop a more efficient and flexible technique, which can also calibrate low-power and/or low-resolution embedded cameras in a robust way. We avoid any central entity in order to ensure scalability and further, we do not require any user-interaction. The algorithm used for distance and angle measurements can be any state-of-the-art computer vision method or even based on a different sensing modality such as infrared.

III. Problem Statement

We consider a set of $n \in \mathbb{N}$ cameras

$$C = \{c_1, ..., c_n\}$$

where a camera c_i has a certain position and orientation which is initially unknown. All cameras operate solely on their respective local coordinate systems within which they define their own position as $(x_i = 0, y_i = 0)$ and their orientation as $\varphi_i = 0°$. This means, that there exist n different coordinate systems in a network of n cameras. In the following we specify the calibration problem in 2D; it can be easily extended to 3D by introducing a further parameter for the height of the camera.

Further, we define a set of $m \in \mathbb{N}$ objects, which can be any targets or key-points used for the sensor measurements

$$O = \{o_1, ..., o_m\}$$

where an object o_j is represented as a 2D point with a certain position (u_j, v_j). Cameras are able to estimate local distances d_{ji} and angles α_{ij} to objects in case they are within the camera's FOV. Since cameras define their own orientation as $\varphi_i = 0°$, the horizontal angle to an object is positive if the object was detected in the right part of the FOV and negative otherwise. The FOV of a camera is illustrated by the gray colored area in Figure 1, whereby d_{ij} is the locally measured distance from camera c_i to object o_j, and α_{ij} is the local angle at which the object was detected by the camera.

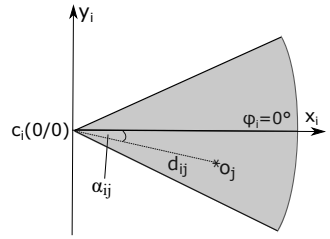

Fig. 1. The FOV of a camera c_i including its local coordinates and orientation and a detected object o_j. The locally measured distance to the object is d_{ij} and the measured angle is α_{ij}.

Two cameras c_i and c_j are considered as neighbors, if they have an overlapping FOV. The overall goal is, that cameras gather a global network view by estimating positions and orientations of all other nodes in the network autonomously. Note, that the global view is constructed w.r.t. a camera's local coordinate system, since we do not have access to any real world coordinates. Since we consider a decentralized network, we distribute the calibration tasks to each camera. For our calibration algorithm we assume following capabilities: i) cameras can communicate with each other, ii) cameras are synchronized and intrinsically calibrated, iii) cameras are capable of object detection and matching and iv) cameras can locally measure the distance and the angle to detected objects.

IV. Network calibration algorithm

We have developed an algorithm called Camera Network Self-Calibration (CaNSeC), which enables simple, autonomous network calibration. The algorithm consists of three parts, namely i) object-based estimation, ii) 1-hop neighbor calibration and iii) network-wide calibration. In the object-based estimation, cameras construct the so-called observation vector υ_{c_i}, which includes information about detected objects. The 1-hop neighbor calibration algorithm calibrates cameras

with overlapping FOVs (considered as neighbors) in a pairwise manner, using the observation vectors. A pairwise calibration results in a localization vector τ_{hi}. This vector is used in the network-wide calibration to carry calibration information through the network to extend calibration to non-neighboring nodes.

Object-based estimation: In the object-based estimation, cameras measure distances and angles to objects. Whenever an object o_j is detected by a camera c_i, the camera extracts an object descriptor f_{ij}, measures distance d_{ij} and horizontal angle α_{ij} to the object and generates a time stamp t_z.

Cameras extract descriptors from detected objects to enable a re-identification of objects at neighboring cameras. Descriptors typically include object characteristics such as color, texture or shape. There is a lot of research in the field of object re-identification, especially on the design of discriminative, descriptive and robust visual descriptors [13]. We will thus make use of existing methods, from which we will choose one with low complexity.

Using all these sensor measurements, cameras construct the following observation vector:

$$\upsilon_{c_i} = (t_z, f_{ij}, d_{ij}, \alpha_{ij})$$

Since cameras assume to have an orientation of $\varphi_h = 0°$, the object's position (u_{ij}, v_{ij}) is defined by the polar coordinates:

$$u_{ij} = d_{ij} \cdot cos(\alpha_{ij})$$
$$v_{ij} = d_{ij} \cdot sin(\alpha_{ij})$$

Whenever a camera has finished the object-based estimation, it broadcasts its observation vector, which will trigger the further calibration process on neighboring cameras. The pseudo code is shown in algorithm 1 and runs on each camera in the network.

Algorithm 1 Object-based estimation algorithm

on object o_j is detected at camera c_i
  generate time stamp t_z
  generate object identifier f_{ij}
  estimate distance d_{ij} to object
  estimate angle α_{ij} to object
  broadcast observation vector υ_{c_i}

1-hop neighbor calibration: 1-hop neighbors are cameras which have an overlapping FOV, being thus directly connected via joint object detections. A joint object detection between two cameras c_i and c_h is defined by the following function

$$joint(o_{ij}, o_{hj}) = \begin{cases} 1, & (f_{ij} - f_{hj} < \varepsilon) \wedge (t_{z_i} - t_{z_h} < \omega) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where f_{ij} and f_{hj} are the object descriptors, t_{z_i} and t_{z_h} are the time stamps at which the objects were detected, and ε and ω are error tolerances depending on the sensor measurement technique and the frame rate, respectively. To be able to determine the difference $(f_{ij} - f_{hj})$ of two object descriptors, a metric needs to be defined depending on which object characteristics are used for the descriptors.

This algorithm calibrates neighbors in a pairwise fashion and starts whenever an observation vector υ_{c_i} is received by a camera. First, the receiving camera c_h determines if $joint(o_{ij}, o_{hj}) = 1$. If so, camera c_h uses the received distance d_{ij} to estimate the position of the sending camera c_i. Therefore, it estimates a circle with radius d_{ij} and center (u_{hj}, v_{hj}) using following equation

$$d_{ij}^2 = (x_i - u_{hj})^2 + (y_i - v_{hj})^2 \quad (2)$$

The circle estimation of camera c_h to locate camera c_i is depicted in Figure 2. After one joint object detection, a camera can thus constrain the neighbor's position to a circle.

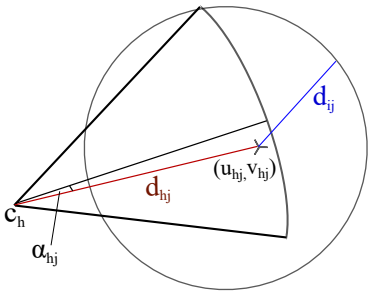

Fig. 2. Position estimate of camera c_h to locate the neighboring camera c_i. Camera c_i lies somewhere on the estimated circle.

With further joint object detections, additional circles are estimated resulting in a system of non-linear equations, which is solved to find the relative position of the neighboring node. Three circles are enough to define a unique solution. Figure 3 shows an example of two cameras and three joint object detections, after which the position of the neighboring camera was estimated.

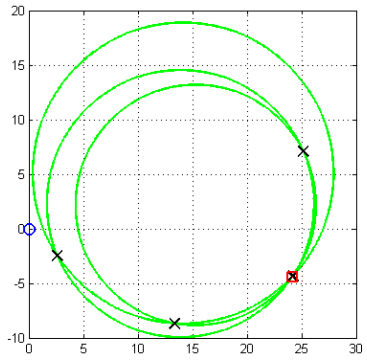

Fig. 3. The estimating camera is depicted by a small blue circle located in the origin. Each large circle represents a joint object detection and the crosses depict all circle intersections. The small red rectangle shows the estimated position of the neighboring camera.

As soon as the neighbor's position is determined, the relative orientation is estimated. Camera c_h calculates the angle β between the neighbor's position (x_i, y_i) and the position of the

object (u_{hj}, v_{hj}). To find the relative orientation of camera c_i, the received angle α_{ij} and the calculated angle β are summed up as shown in the following equation

$$\varphi_i = \arctan \frac{v_{hj} - y_i}{u_{hj} - x_i} \cdot \frac{180}{\pi} + \alpha_{ij} \tag{3}$$

The angle estimation of camera c_h to orient camera c_i is depicted in Figure 4. β is the angle of interest and φ_{ic_h} is the unknown.

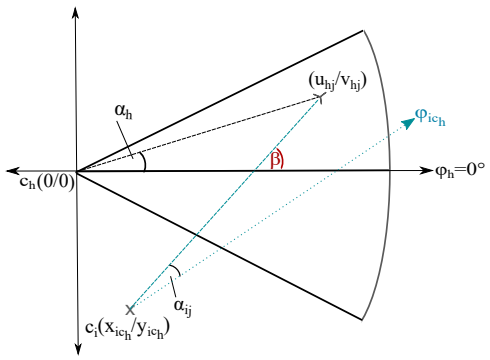

Fig. 4. Angle estimation of camera c_h to orient camera c_i. β is the angle which needs to be calculated and φ_{ic_h} is the unknown.

After a 1-hop neighbor localization has been finished, a localization vector τ, including the relative position (x, y) and orientation φ of the neighbor w.r.t. the local coordinate system of the camera which performed the localization, is broadcasted. The localization vector τ_{hi} is thus constructed by camera c_h, to inform other neighbors about camera c_i

$$\tau_{hi} = (x_{ic_h}, y_{ic_h}, \varphi_{ic_h})$$

The pseudo code of the 1-hop neighbor calibration is shown in algorithm 2 and runs, as algorithm 1, on each camera in the network.

Algorithm 2 1-hop neighbor calibration algorithm

On υ_{c_i} is received at camera c_h
  if object o_j was detected **then**
    if $t_{z_i} - t_{z_h} < \omega$ and $f_{ij} - f_{hj} < \varepsilon$ **then**
      estimate position of sender node using equation 2
      estimate orientation of sender node using equation 3
      broadcast localization vector τ_{hi}

Network-wide calibration: This part of the algorithm calibrates the whole network by exchanging localization vectors between multi-hop neighbors. Assume camera c_g has already localization information of camera c_h and wants to gather information about camera c_i using the received localization vector τ_{hi} of camera c_h. If the sending camera c_h has been localized already, c_g knows the relative position of c_h and can use it to transform the received coordinates of c_i to its own local coordinate system. This is done by rotating and translating the received coordinates:

$$\begin{pmatrix} x_{ic_g} \\ y_{ic_g} \end{pmatrix} = R \cdot \begin{pmatrix} x_{ic_h} \\ y_{ic_h} \end{pmatrix} + T \tag{4}$$

where R is the rotation matrix, T the translation vector and (x_{ic_h}, y_{ic_h}) are the coordinates of camera c_i w.r.t. to the coordinate system of camera c_h. The rotation matrix R is defined using the known orientation φ_{hc_g} of camera c_h, and the translation vector T using the known coordinates (x_{hc_g}, y_{hc_g}) of camera c_h:

$$R = \begin{bmatrix} \cos(\varphi_{hc_g}) & -\sin(\varphi_{hc_g}) \\ \sin(\varphi_{hc_g}) & \cos(\varphi_{hc_g}) \end{bmatrix}, T = \begin{pmatrix} x_{hc_g} \\ y_{hc_g} \end{pmatrix}$$

The orientation φ_{ic_g} of camera c_i is determined by the sum of the known orientation φ_{hc_g} and the received orientation φ_{ic_h}:

$$\varphi_{ic_g} = \varphi_{hc_g} + \varphi_{ic_h} \tag{5}$$

Figure 5 depicts the process of transforming the coordinates of camera c_i from the coordinate system of camera c_h to that of camera c_g. The translation vector is depicted on the left side of the figure and the rotation is depicted on the right side.

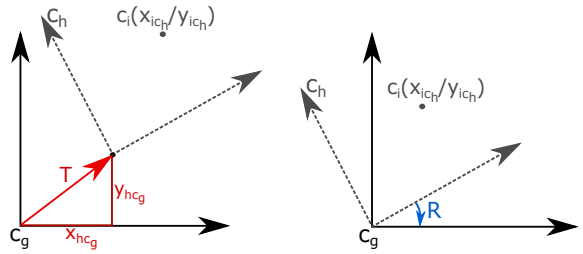

Fig. 5. Translation (left) and rotation (right) to transform the coordinates of camera c_i from the coordinate system of camera c_h to that one of camera c_g.

If c_h has not been localized yet, c_g postpones the transformation of c_i until the position of c_h is known and can be used for the transformation. The network-wide calibration algorithm thus calibrates the whole network step by step on every camera. The last paragraph in section VI discusses how to save resources by executing the algorithm on only one node.

The only requirement to achieve a complete network calibration, is a connected network. The network-wide calibration does not need a fully connected network. However, the higher the node degree of a network, the faster and the more precise the calibration. In case of a non-connected network, each cluster of cameras that is connected is calibrated. The pseudo code of the network-wide calibration is shown in algorithm 3.

Algorithm 3 network-wide calibration algorithm

On τ_{hi} is received at camera c_g
  if the camera c_i has not been localized yet **then**
    if the sender c_h has already been localized **then**
      transform the received coordinates of c_i to own local coordinate system using equations 4 and 5
      broadcast localization vector τ_{gi}
    else
      postpone transformation until c_h is known

V. Calibration with Inaccurate Measurements

Since sensor measurements cannot be assumed to deliver accurate results, error modeling and handling is important to

achieve a robust calibration. In order to compensate errors in sensor measurements and computer vision, we rely on more than three joint object detections and we change our geometric model such that it can handle those errors.

Sensor error model: Distance measurements are superimposed with a uniformly distributed error within the limits $[-\rho,+\rho]$. The limits of the uniform distribution increase with increasing distance to objects, since object identification and distance measurements become typically less reliable the further objects are away from the sensor. We therefore model ρ as a function of the distance:

$$\rho = f(d) = k \cdot d \tag{6}$$

Error handling: The inaccurate sensor measurements introduce an uncertainty in the object-based estimation, which results in following changes in the observation vector:

$$\upsilon_{c_i} = (t_z, f_{ij}, d_{ij} + \eta, \alpha_{ij})$$

η is a random variable, which is used to generate an uniformly distributed uncertainty around the true distance as shown in Figure 6.

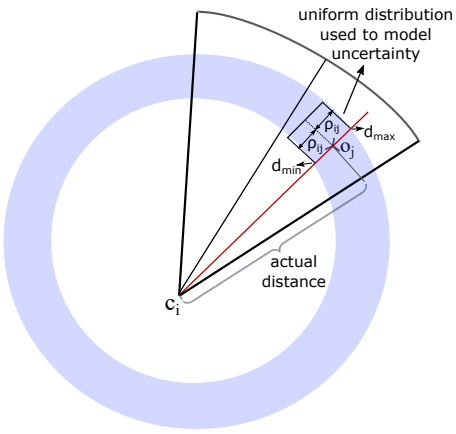

Fig. 6. An error ring following a uniform distribution around the actual distance value.

Using the new observation vector, distance estimations to objects result in a ring instead of a circle (see Figure 6). The object lies somewhere in the colored area of the estimated ring. Thus, estimating the position of a neighboring camera results in an area instead of a single point. We express this area by its center and its size.

Figure 7 shows the position estimation of camera c_h to locate camera c_i after 2 joint object detections and thus two estimated error rings. The intersection areas of all rings are calculated in order to find the position of the neighboring camera. Camera c_i is located somewhere in the striped areas depicted in Figure 7.

Intersecting multiple error rings results in one area which is contained in every ring. The center of this area is then chosen as the neighbor's position (x_{est}, y_{est}). The pseudocode of the 1-hop neighbor calibration considering the sensor error is shown in algorithm 4, where the user-defined parameter K is the number of rings required to trigger the position estimation. The more rings the more precise the solution, but the more joint object detections are required. If e.g., $K = 6$, a camera waits for 6 joint object detections (and thus rings) per neighbor before it decides for the position.

Algorithm 4 1-hop neighbor calibration algorithm

```
On υ_ci is received at camera c_h
  if object o_j was detected then
    if t_zi − t_zh < ω and f_ij − f_hj < ε then
      estimate new error ring
      intersect it with all previously estimated rings
      if number of rings > K then
        find intersection area which lies on every ring
        use area center as estimated position of camera c_i
        estimate orientation of camera c_i
        broadcast localization vector τ_hi
```

Other algorithms which model uncertainties in localization for wireless sensor networks can be found in [14] and [15].

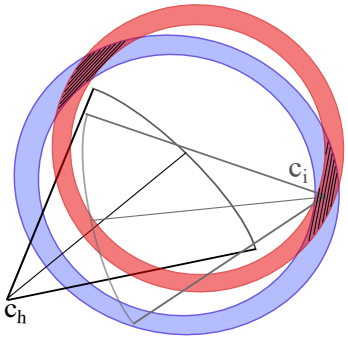

Fig. 7. Two error rings estimated by camera c_h to locate camera c_i. The ring intersection areas are the possible locations of camera c_i and are depicted by black stripes.

VI. Evaluation

In this section we present the results of our evaluations, which have been achieved by our visual sensor network simulator. We have developed this simulator using .NET and Windows Forms to enable easy and fast development and testing of algorithms. So far we 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. [3]. Further, we can simulate objects which are modeled as 2D points with a specific location (u_j, v_j).

For the evaluation, we measured the accuracy of estimated positions and orientations, the number of sent and received messages, the duration of the calibration process and the error propagation through the network. For that purpose we simulated the following three network setups:

- Scenario 1: 3 cameras, average node degree of 2
- Scenario 2: 30 cameras, average node degree of 4
- Scenario 3: 30 cameras, average node degree of 2

In the first two scenarios each camera has multiple neighbors and the networks aim to cover a certain area like a hall. The cameras of the second scenario were placed in a way such that they could cover a corridor or a street. The first evaluations were done using a noise gradient of $k = 0.1$, triggering the ring intersection after $K = 6$ joint object detections per each pair of cameras. Objects were randomly added to the scenarios such that each camera could see at least 2 of them.

Figure 8 shows scenario 1 and the position estimation results.

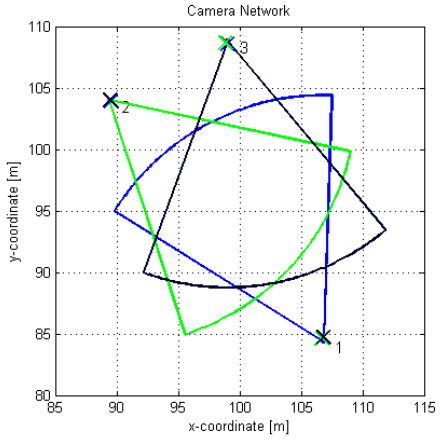

Fig. 8. The calibration results of a network of three cameras. The crosses depict the estimated positions and the circular sectors the ground-truth FOVs.

The colored circular sectors depict the ground-truth FOVs of all cameras, and the small colored crosses are the positions which were locally and independently computed by the cameras and transformed to the world coordinate system after the simulation. The small crosses are thereby the localization results of the cameras depicted in same color (e.g., green crosses are the estimations of camera 2). The estimated positions are very close to the real ones (detailed accuracy measurements will be presented later). Scenarios 2 and 3 and their position estimation results are shown in Figure 9 and Figure 10, respectively.

Figure 11 shows the evaluation results of the accuracy measurements. The upper plot depicts the error of the estimated positions and the lower plot the error of the estimated orientations. We normalized the position estimation error in order to provide a meaningful comparison between different network setups and sizes. The normalized position error was calculated using the root-mean-squared error (RMSE) and dividing the result by the diameter of the network r_{net} as shown in equation 7.

$$RMSE_{pos} = \frac{1}{r_{net}} \sqrt{\frac{\sum_{k=1}^{n} (x_k^{true} - x_k^{est})^2 + (y_k^{true} - y_k^{est})^2}{n}} \quad (7)$$

where n is the number of cameras in the network, (x_k^{true}, y_k^{true}) is the ground truth position of camera c_k, (x_k^{est}, y_k^{est}) is the estimated position and r_{net} is the diameter of the network.

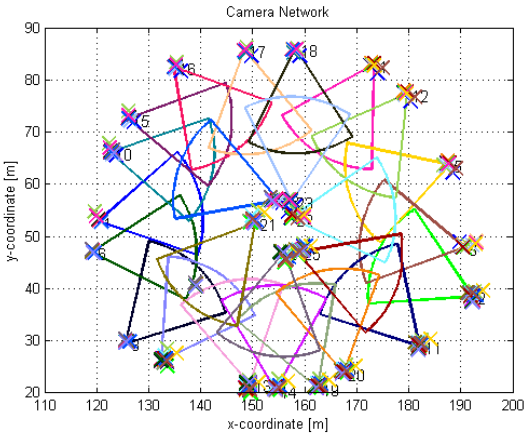

Fig. 9. Camera networks of 30 cameras placed in a way such that they could e.g., cover a large area. The crosses depict the estimated positions and the sectors the ground-truth FOVs.

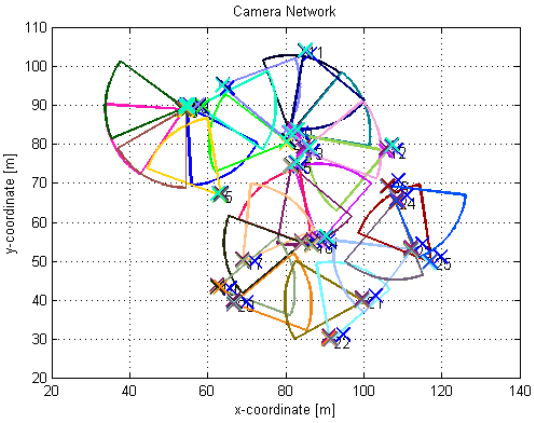

Fig. 10. Camera networks of 30 cameras placed in a way such that they could e.g., cover a corridor. The crosses depict the estimated positions and the sectors the ground-truth FOVs.

The orientation error was computed using the RMSE, without any normalization as shown in equation 8.

$$RMSE_{orient} = \sqrt{\frac{\sum_{k=1}^{n} (\varphi_k^{true} - \varphi_k^{est})^2}{n}} \quad (8)$$

where φ_k^{true} is the true orientation of camera c_k and φ_k^{est} the estimated orientation. The diagrams in Figure 11 show that the large network with small node degree performs worst. This is because the error is accumulated when executing algorithm 3 for the network-wide calibration. For the other two networks, the positioning error stays below 0.2 and the orientation error stays below 22° for noise gradients of $k < 0.25$. Since we normalize the positioning error by the network diameter, this means that the error does not exceed 20% of the network diameter size.

The communication costs are presented in Figure 12. The upper plot shows the average number of sent messages per node and the lower plot the average number of received

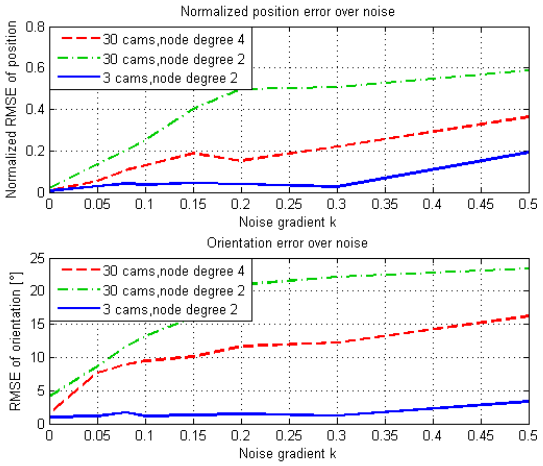

Fig. 11. Normalized RMSE of position and orientation over varying noise level for different network sizes and node degrees.

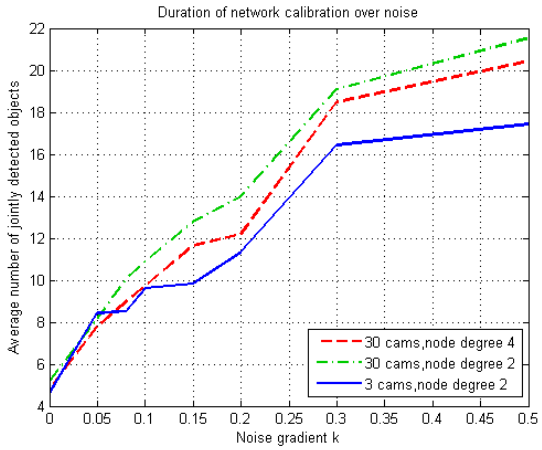

Fig. 13. Simulation steps required for the whole network to calibrate for different network sizes and node degrees.

messages per node required to calibrate the whole network. The large networks with 30 cameras require per node for a noise gradient of $k = 0.05$ on average 35 sent and 870 received messages, while the small network requires 9 sent and about 19 received messages. The communication costs are high since cameras broadcast every observation and localization vector. These broadcasts significantly increase especially the received messages. We will discuss improvements and communication cost savings in the last paragraph of this section.

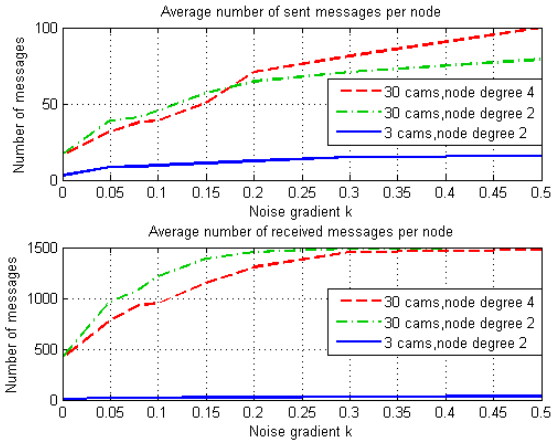

Fig. 12. Average number of sent and received messages per node until the whole network calibration was finished for different network sizes and node degrees.

Figure 13 shows the average number of jointly detected objects per camera pair, required to calibrate the whole network. As the graph shows, there is no major difference for the different types of networks, which makes sense since the calibration process is based on pairwise camera interaction, which does not change with changing network setups. For noise gradients of up to $k = 0.2$, none of the scenarios requires camera pairs to detect more than 14 objects jointly. A realistic error scenario of $k = 0.05$ would require camera pairs in each scenario to detect approximately 8 objects jointly.

We simulated an additional scenario in order to take a closer look on the error propagation through the network (see Figure 14). In this scenario, the network topology corresponds to a single line without any circle. The network consists of 10 cameras, with an average node degree of 2. The FOV of camera c_1 overlaps only with that of camera c_2, which means that almost all cameras are localized using the network-wide calibration algorithm. Figure 14 shows the positioning results of camera c_1 compared to the ground truth. The FOVs of all 10 camera, depicted in different colors, represent the ground truth. The blue crosses are the camera positions which were locally estimated at camera c_1, whereby camera c_2 was positioned using the 1-hop neighbor calibration and cameras $c_3 - c_{10}$ were positioned using the network-wide calibration algorithm.

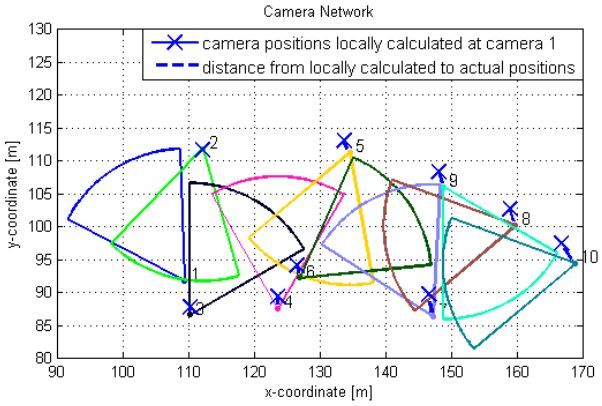

Fig. 14. Ground truth network compared to the estimated position results of camera 1 (blue crosses).

To understand the process better, we take a look at camera c_3, which can localize cameras c_2 and c_4 directly since they are 1-hop neighbors. Camera c_3 broadcasts the localization vectors for camera c_2 and c_4, which means that camera c_2 learns the

position of camera c_4 through translation and rotation. Camera c_2 will further communicate the newly learned position to its neighbors, which means that now also camera c_1 can estimate the position of camera c_4 since it knows camera c_2 already (pre-requisite to enable translation and rotation). In this manner, calibration information is carried through the network. However, the errors of every single camera are accumulated during this process. This is what we can also see from Figure 14. The error for cameras closer to camera c_1 is smaller than for cameras further away. For this simulation scenario we used a noise gradient of $k = 0.1$ and we triggered the ring intersection after $K = 6$ joint object detections.

Complexity and improvements: CaNSeC is a resource efficient algorithm, which solely relies on geometric calculations including trigonometric functions. Communication costs are expected to be low since observation and localization vectors are only required within the neighborhood of nodes. In our simulation study we have not limited the data exchange to this neighborhood, but used broadcasts for the data exchange. As a consequence we achieved high communication costs. Our next steps will thus include significant reduction of messaging. This will be done by introducing unicast messages in algorithms 3 and 4 and by splitting camera FOVs into several parts, in which newly created observation vectors are broadcasted only with a certain probability. Introducing unicast messages does not reduce the number of communicated messages in the case of a wireless medium, however, it reduces processing costs, since cameras which are not the intended receivers do not need to process the message. In a wired medium, unicast messages would significantly decrease the number of received messages on all cameras.

There is also a significant potential of reducing communication and processing costs in the network-wide calibration algorithm since a global network view can be constructed on just one node. This means, that localization vectors are communicated only to one specific node. It would also be possible to introduce a further step before algorithm 1, which takes care of neighbor discovery, which would allow nodes to send observation vectors only to neighbors. Having these simple extensions, communication and processing costs will only increase with increasing node degree but not with increasing network size.

In order to improve the calibration results for large networks and networks with a small node degree, we will introduce an online adaptation and refinement of estimated positions in algorithm 4.

VII. Conclusion

In this paper we presented an autonomous and efficient VSN network calibration algorithm, called CaNSeC, which delivers promising results. Cameras learn relative positions and orientations of each other in a fully decentralized way. Summarized, we can say that CaNSeC turned out to be a fast and resource efficient method to calibrate distributed camera networks without any central entity and without user-interaction. With further extensions, the algorithm will use even less messages and will refine estimated positions in order to deliver more precise results. We believe that a major factor of future networks will be usability and scalability since networks increase and become part of our everyday life.

Acknowledgements

Many thanks to Matthias Weyrer for his valuable ideas and comments. This work is supported by the research initiative 'Intelligent Vision Austria' with funding from the Austrian Federal Ministry of Science, Research and Economy and the Austrian Institute of Technology.

References

[1] J. Simonjan and B. Rinner, "Autonomous, lightweight calibration of visual sensor networks with dense coverage," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communication Workshops*. IEEE, 2016, pp. 1–4.

[2] V. P. Munishwar and N. B. Abu-Ghazaleh, "Coverage algorithms for visual sensor networks," *ACM Transactions on Sensor Networks*, vol. 9, no. 4, pp. 1–36, 2013.

[3] 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.

[4] T. Ellis, D. Makris, and J. Black, "Learning a multi-camera topology," in *Proceedings of the Joint IEEE Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, 2003, pp. 165–171.

[5] B. Rinner, L. Esterle, J. Simonjan, G. Nebehay, R. Pflugfelder, G. F. Domínguez, and P. R. Lewis, "Self-aware and self-expressive camera networks," *Computer*, vol. 48, no. 7, pp. 21–28, 2015.

[6] A. Van Den Hengel, A. Dick, and R. Hill, "Activity topology estimation for large networks of cameras," in *Proceedings of the IEEE International Conference on Video and Signal Based Surveillance*. IEEE, 2006, pp. 44–49.

[7] F. Lv, T. Zhao, and R. Nevatia, "Camera calibration from video of a walking human." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1513–1518, 2006.

[8] J. Liu, R. T. Collins, and Y. Liu, "Surveillance camera autocalibration based on pedestrian height distributions," in *Proceedings of the British Machine Vision Conference*, 2011, p. 144.

[9] H. Possegger, M. Rüther, S. Sternig, T. Mauthner, M. Klopschitz, P. M. Roth, and H. Bischof, "Unsupervised calibration of camera networks and virtual ptz cameras," in *Proceedings of the 17th Computer Vision Winter Workshop*, 2012.

[10] A. Ortega, M. Silva, E. H. Teniente, R. Ferreira, A. Bernardino, J. Gaspar, and J. Andrade-Cetto, "Calibration of an outdoor distributed camera network with a 3d point cloud," *Sensors*, vol. 14, no. 8, pp. 13 708–13 729, 2014.

[11] F. Yin, D. Makris, S. A. Velastin, and T. Ellis, "Calibration and object correspondence in camera networks with widely separated overlapping views," *IET Computer Vision*, vol. 9, no. 3, pp. 354–367, 2015.

[12] J. Guan, F. Deboeverie, M. Slembrouck, D. Van Haerenborgh, D. Van Cauwelaert, P. Veelaert, and W. Philips, "Extrinsic calibration of camera networks based on pedestrians," *Sensors*, vol. 16, no. 5, p. 654, 2016.

[13] A. Bedagkar-Gala and S. K. Shah, "A survey of approaches and trends in person re-identification," *Image and Vision Computing*, vol. 32, no. 4, pp. 270–286, 2014.

[14] J. Fang, D. Duncan, and A. S. Morse, "Sequential localization with inaccurate measurements," in *Localization Algorithms and Strategies for Wireless Sensor Networks: Monitoring and Surveillance Techniques for Target Tracking*. IGI Global, 2009, pp. 174–197.

[15] M. L. Sichitiu, V. Ramadurai, and P. Peddabachagari, "Simple algorithm for outdoor localization of wireless sensor networks with inaccurate range measurements," in *Proceedings of the International Conference on Wireless Networks*, 2003, pp. 300–305.