Self-calibration and Cooperative State Estimation in a Resource-aware Visual Sensor Network

Jennifer Simonjan[†], Melanie Schranz[*], Bernhard Rinner[†]
[*]Lakeside Labs, Austria, Email: schranz@lakeside-labs.com
[†]Alpen-Adria-Universität Klagenfurt, Austria, Email: Jennifer.Simonjan@aau.at, Bernhard.Rinner@aau.at

Abstract—**In this paper we present an algorithm, which enables distributed visual sensor networks to autonomously calibrate the network and dynamically build clusters to achieve cooperative object tracking based on state estimation. A main focus is thereby on resource-awareness and -efficiency, since we aim for low-power embedded smart camera networks. We do not require any human intervention or a-priori information about the network topology to achieve calibration and tracking. Camera nodes first estimate relative positions and orientations and then use the common coordinate system to enable cooperative state estimation. For that purpose, cameras dynamically build clusters depending on their available resources. New nodes joining the network are discovered and failing nodes do not prevent others from their tasks. Compared to other methods, our approach is not only able to handle sensor measurement errors but also faulty camera positions gathered during the network calibration process.**

I. Introduction

Visual Sensor Networks (VSNs) consist of a set of static smart cameras, each having processing and communication capabilities on-board. Typical VSNs are spatially distributed, composed of a large number of nodes and often deployed in environments without any infrastructure. Each node poses limitations in processing, storage and communication capabilities due to the local energy reservoir. Thus, resource-awareness plays a key role within these networks.

In this paper we introduce an algorithm for VSNs, which is capable of autonomous network calibration and resource-aware, cooperative target tracking. For that purpose, we integrate state estimation based on dynamic clustering [1] with network calibration [2] and can therefore relax the assumption of a known absolute coordinate system. Our approach works in a fully decentralized manner, without requiring any human interaction or a-priori network information. The main design goals are the following:

- self-calibration, to ease the network management,
- adaptability, to handle dynamic environments together with changing network topologies,
- resource efficiency, to achieve a long lifetime for low-power embedded sensor networks.

Most existing work for resource-aware dynamic clustering assumes a known configuration of the cameras in the network [3]–[6]. Approaches for autonomous network calibration typically focus on computer vision solutions [7]–[9], which are computationally much more complex. There is a lot of research available on how to estimate either camera or object positions having erroneous measurements. The idea of this paper was to estimate the camera positions as well as the object positions to enable a self-configuring network and to avoid the need for an a-priori known ground plane. All position estimates are thereby influenced by measurement errors and estimation errors of each other. The results show that a cooperative state estimation of object positions works well even though having faulty camera position estimates.

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 approach for network calibration and object tracking. The evaluation results are presented in section V. We conclude and outline future work in section VI.

II. Related work

There is much research going on in the field of cooperative state estimation and multi-target tracking. In typical collaborative tracking applications, camera networks are assumed to be externally calibrated, since the common ground plane is required to perform joint state estimation. A calibrated network provides thus the basis for cooperative tracking.

An example approach for multi-target tracking in self-configuring pan-tilt-zoom (PTZ) camera networks was presented by Soto et al. [3]. Their overall goal is to achieve cooperative tracking using a Kalman-Consensus filter, while selectively focusing on specific target features with higher resolution.

A similar approach was introduced by Liu et al. [4]. Whenever an object is detected by several cameras, they form a cluster to cooperatively track the object. A main advantage of their approach is, that an optimization-based algorithm selects only a subset of cameras as cluster members, rather than all cameras which can see the object, in order to save resources.

SanMiguel and Cavallaro [5] proposed another framework, which creates camera coalitions for collaborative object tracking in resource-constrained networks. Coalition heads are, like in our approach, selected dynamically. The authors came up with a decentralized negotiation scheme to allocate resources to coalitions over time.

Another framework for distributed state estimation, called information-weighted Consensus Filter (ICF), was proposed by Kamal et al. [6]. It additionally incorporates the cross-covariances between the individual states of the nodes into the distributed estimation. Such a consensus filtering scheme can guarantee convergence to the optimal centralized estimate.

None of the approaches mentioned above allows cameras to autonomously estimate the network topology, which means that a known global ground plane is assumed. This is also true for the many other cooperative tracking approaches which can be found in literature.

To establish a global coordinate system, network calibration algorithms typically rely on cost-intensive computer vision methods such as vanishing line estimations [8], 3D position estimations [7] or 3D point clouds [9]. We require a simple, efficient network calibration method to ensure resource-efficiency as well as operability on low-resolution embedded cameras.

Distributed state estimation and localization can also be found in the field of robotics. In decentralized, cooperative Simultaneous Localization and Mapping (SLAM) approaches, mobile robots jointly build a map of the environment while localizing themselves within the map. Multiple robots usually collect local sensor measurements to landmarks, exchange them and perform a cooperative state estimation [10], [11]. SLAM algorithms typically work in environments with mobile sensors and static targets, while we focused on static sensors and mobile targets. However, the localization and state estimation process is similar.

In our work, we were interested in the performance of cooperative object tracking in case the cameras estimate the ground plane coordinate system autonomously, introducing errors for camera positions. This means, our state estimation algorithm has to handle inaccurate object measurements as well as erroneous camera positions.

III. Problem Formulation

In this paper, we consider a set of $n \in \mathbb{N}$ smart cameras $C = \{c_1, ..., c_n\}$, whereby at least some of them have an overlapping field of view (FOV). Two example network setups are depicted in figure 1. The network in the upper plot is connected in terms of overlapping FOVs. The lower plot shows two disconnected sub-networks, namely cameras $c_1, \dots, c_4$ and cameras $c_5, \dots, c_7$.

Each camera c_i in the network has an initially unknown position and orientation and assumes itself to be the origin of the coordinate system with an orientation of 0°. This means, that every camera operates on its own local coordinate system. Two cameras c_i and c_h are considered as neighbors, if they have an overlapping FOV. We further define a set of $j \in \mathbb{N}$ mobile objects $O = \{o_1, ..., o_j\}$, which are typically humans in camera surveillance or tracking applications.

The overall task of the VSN is to let cameras cooperatively track one or more specific objects through the network by jointly estimating the object positions. To enable a joint state estimation, cameras have to operate on the same coordinate system, which they have to establish first. This means, the starting point of our system is right after the deployment of a camera network, when cameras do not have any topology information. The output of the system includes a common coordinate system established by the cameras as well as the estimated tracks of objects which moved through the network.

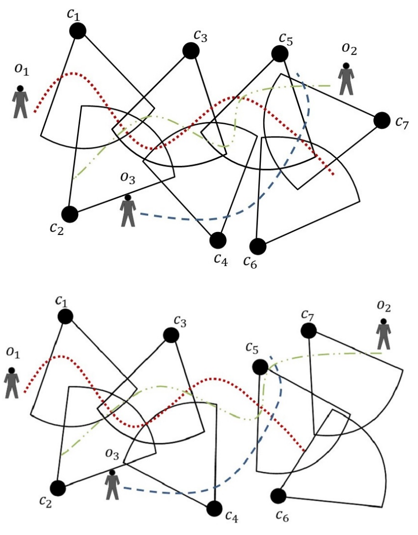


Fig. 1. Two possible camera network setups, whereby one network is connected in terms of overlapping FOVs and the other network is split into two connected networks.

For that purpose, we require cameras to be synchronized in time and to be able to locally perform certain processing tasks such as object detection, re-identification and state estimation. Further, we assume cameras to be able to measure local distances and angles to objects within their FOVs. Figure 2 shows the FOV of a camera c_i within its own local coordinate system and with the local distance and angle measurements to an object o_j.

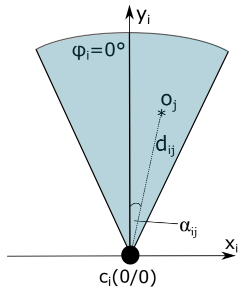


Fig. 2. The model of a FOV of a camera c_i within its own local coordinate system. d_{ij} and α_{ij} are the locally measured distance and angle to the object. φ_i is the orientation of the camera, which is 0 in the local coordinate system.

In a nutshell, our approach performs the following steps:

1. calibrate cameras relative to each other wrt. their local coordinate systems
2. establish common coordinate systems within connected sub-networks
3. perform resource-aware cooperative object tracking

IV. Algorithm

We have developed an algorithm which calibrates camera networks autonomously to afterwards enable cooperative tracking of objects utilizing state estimation. For all camera pairs which have detected the same object in their FOVs, the cameras first check whether calibration is necessary and perform then the cooperative object tracking.

As we assume that at least some cameras have overlapping FOVs, an object may be seen by multiple cameras at the same time. All cameras that can see the object o_j at the same time belong to the same tracking cluster Ω^j, within which only one camera c_i is responsible of estimating the object's position in order to save resources. This camera is thus referred to as tracking cluster head $c_i^h \in \Omega^j$ of object o_j. To estimate the object position, the tracking cluster head c_i^h collects object observations of all tracking cluster members $c_{k\neq i}^m \in \Omega^j$ and performs a cooperative state estimation using the Kalman Consensus Filter proposed by Ding et al. [12].

To enable a joint state estimation, all object estimations must be performed in the same coordinate system. A common coordinate system can only be established within a network of connected cameras, since cameras estimate relative positions based on jointly detected events. We therefore may need multiple coordinate systems, one for each connected sub-network. In the lower plot of figure 1 for example, cameras $c_1,\ldots,c_4$ as well as cameras $c_5,\ldots,c_7$ establish two independent coordinate systems. The network calibration approach introduced in [2], allows each camera in a connected network to establish a global network view wrt. its local coordinate system. However, to perform the cooperative tracking, the cameras need to agree on one common coordinate system. Therefore, we introduce a further cluster, called coordinate system cluster Ψ^s, whereby s differentiates between the connected sub-networks. These clusters also have a cluster head $c_i^h \in \Psi^s$ and cluster members $c_{k\neq i}^m \in \Psi^s$. All cluster members agree during the calibration process on the coordinate system of the cluster head, which means that the final coordinate system has its origin at the position of the cluster head.

Calibration of cameras is done only if they have not localized each other yet (when they detect joint objects for the first time), while tracking clusters change dynamically with the moving objects. Only cameras which can see the same object at the same time belong to a common tracking cluster. The head and the members of tracking clusters thus change as the object moves through the network. Cameras may also belong to multiple tracking clusters in case they can observe multiple objects at the same time.

In the following subsections we will describe the details of the algorithm, which is split into three major parts: object observation, network calibration and state estimation.

Object observation

This algorithm is triggered whenever an object o_j is detected by a camera c_i, to construct an object observation vector O_{ij}. The vector includes a time stamp t_x, the utility λ_i, an object descriptor f_{ij}, the locally measured distance d_{ij} and angle α_{ij} to the object and the flags ς_i^h and ς_i^m, which indicate whether the camera is a coordinate system head, member or none of these.

The utility λ_i comprises information about the resources of a camera and the detection/tracking confidence, and is used to elect both, coordinate system and tracking cluster heads.

The object descriptor f_{ij} is used to enable re-identification of objects at neighboring cameras. Visual descriptors usually include object characteristics such as texture, color or shape to enable robust re-identification [13].

The flags ς_i^h and ς_i^m indicate not only if a camera c_i is a coordinate system head or a member, but also if there already exists a common coordinate system or not. If either of them is $true$, this indicates that there is already a head $c_i^h \in \Psi^s$, whose coordinate system should be used to establish a common network view. If both flags are $false$ there is no coordinate system head and the cameras still need to agree on one. Using all these measurements the following object observation vector is defined as:

$$O_{ij} = (t_x, \lambda_i, f_{ij}, \alpha_{ij}, d_{ij}, \varsigma_i^h, \varsigma_i^m)$$

After the vector O_{ij} has been determined, the camera waits a random back-off time and then broadcasts it. The random back-off time is required, to reduce the probability that multiple camera pairs exchange their first object observation at the same time, which would lead to a simultaneous election of coordinate system heads.

Receiving an observation vector triggers the calibration or tracking process on neighboring cameras. The pseudo code of the object observation is shown in algorithm 1 and runs on every camera in the network.

Algorithm 1 Object observation algorithm

```
On object $o_j$ is detected at camera $c_i$
  do construct object observation $O_{ij}$:
        generate timestamp $t_x$, utility $\lambda_i$ and descriptor $f_{ij}$
        estimate local angle $\alpha_{ij}$ and distance $d_{ij}$ to object
  if camera $c_i$ is coordinate system head then
     set coordinate system head flag $\varsigma_i^h$ to $true$
  else if camera $c_i$ is coordinate system member then
     set coordinate system member flag $\varsigma_i^m$ to $true$
  else
     set both flags $\varsigma_i^h$ and $\varsigma_i^m$ to $false$
  wait random back-off time
  broadcast object observation $O_{ij}$
```

Network calibration

The network calibration algorithm calibrates cameras of connected networks to one common coordinate system. This is not a real-world coordinate system since we do not have access to ground plane coordinates, rather it is established wrt. the cluster head's local coordinate system. The network calibration is triggered in the following two cases: Either an object observation has been received from a camera that has not been localized yet or localization information about a multi-hop neighbor has been received.

Assume camera c_i receives an object observation O_{hj} of another camera c_h for object o_j. If camera c_h has not been localized yet, the calibration process will start. First of all, the coordinate system flags ς_i^h, ς_i^m, ς_h^h and ς_h^m are checked. If any of the flags is $true$, this indicates that there is already a common coordinate system. If all flags on both cameras are *false*, there is no coordinate system to which camera c_i and c_h belong to. In this case, the cameras decide, based on the utility, for a coordinate system head before they start the calibration process.

After checking the coordinate system flags, the cameras start to calibrate with each other in a pairwise manner. Each camera estimates the relative position and orientation of the neighbor within its local coordinate system. The estimation is based on geometric constraints and can be found in [2]. This means, there are two different coordinate systems after the pairwise calibration, one on each camera.

Whenever a camera c_h finished the localization of a neighboring camera c_i, it informs others about camera c_i by broadcasting a localization vector τ_{hi}. The localization vector includes the relative position and orientation of camera c_i wrt. the local coordinate system of camera c_h and enables cameras which do not overlap with camera c_i to localize it.

Assume camera c_g receives the localization vector τ_{hi} of camera c_h including the localization data of camera c_i. In case camera c_g has already localized the sending camera c_h, it can transform the coordinates of camera c_i from the coordinate system of camera c_h to its own local coordinate system. This transformation is achieved by translation and rotation of the coordinates. In this way, all cameras within a connected network can be calibrated.

Whenever the coordinate system head $c_i^h \in \Psi^s$ localizes another camera c_g, it informs c_g about its estimated position. Since coordinate system heads are located in the origin of the common coordinate systems, the common system is the same as their local system. This means, that local position estimates done by coordinate system heads are those which are used for the common coordinate system and thus for the cooperative state estimation.

Algorithm 2 shows the network calibration algorithm, which also runs on every camera within the network. The algorithms were realized as threads and run therefore in parallel on the cameras.

Algorithm 2 Network calibration algorithm

```
On object observation $O_{hj}$ is received at camera $c_i$ from $c_h$
  if object $o_j$ has also been detected then
    if sender $c_h$ has not been localized yet then
      if all flags $\varsigma_i^h$, $\varsigma_i^m$, $\varsigma_h^h$ and $\varsigma_h^m$ are false then
        compare camera utilities $\lambda_i$ and $\lambda_h$
        if $\lambda_i > \lambda_h$ then
          $c_i$ becomes coordinate system head
        else
          $c_i$ becomes coordinate system member
      estimate location and orientation of sender $c_h$
      if sender $c_h$ was successfully localized then
        broadcast localization information
        if $\varsigma_i^h$ is $true$ then
          inform $c_h$ about its estimated coordinates

On localization vector $\tau_{hg}$ is received at camera $c_i$
  if the camera $c_g$ has not been localized yet then
    if the sender $c_h$ has already been localized then
      transform the received coordinates of $c_g$ to own coordinate system
      broadcast localization information $\tau_{ig}$
      if $\varsigma_i^h$ is $true$ then
        inform $c_g$ about its estimated coordinates
```

State estimation

Cooperative tracking can be performed by two or more cameras, if they operate on the same coordinate system and observe the same object. Thus, the prerequisite to start the tracking process is that the cameras have localized each other within the common coordinate system.

Thus, cooperative tracking is triggered if an object observation is received and the sending camera has already been localized. In this case, a tracking cluster Ω^j is formed by electing a cluster head $c_i^h \in \Omega^j$ based on the utilities λ of the cameras. Afterwards, the cluster head initiates an auction to collect state vectors s_{kj} from cluster members $c_k^m \in \Omega^j$. These state vectors include the estimated object position and the camera utility: $s_{kj} = (x_j, y_j, \lambda_k)$.

The estimated object position (x_j, y_j) is thereby determined within the common coordinate system which was established during the calibration phase. The utility λ_k is required to enable a dynamic cluster head election and adaptation. Whenever the utility of a cluster member exceeds that one of the cluster head, their roles change. The election is based on auctions as described in [1].

The cluster head receives the state vectors of all cluster members and performs the state estimation. Tracking clusters dynamically change with moving objects, since only cameras which can see the same object as the cluster head will answer to the auction. The state estimation process is shown in algorithm 3 and runs, as the other two algorithms, on every camera.

V. Evaluation

The proposed algorithm is evaluated via simulation studies. For that purpose, we extended the VSN simulator [14], which allows the algorithms including object detection, calibration and state estimation to run locally on each camera.

Scenario: Figure 3 shows the simulation scenario we used. We simulated a building with multiple rooms, whereby one object moved through four of them. The path depicted by the red line shows the object movement through the rooms 1, 2, 4 and 14. Sensor measurements are performed by the cameras every 3 seconds. The duration of the measurements differs for each room, depending on the path length of the object, e.g., $88,67m$ for room 14 and $28,63m$ for room 2, whereby the object moves $0.1m$ per second. The whole path including all rooms and gangways has a length of $185,8m$.

Algorithm 3 State estimation algorithm

```
On object observation $O_{hj}$ is received at camera $c_i$ from $c_h$
  if sender $c_h$ is localized already then
    if object $o_j$ has also been detected then
      if camera $c_i$ is cluster head then
        initiate auction
        receive state vectors $s_{kj}$ of cluster members
        perform state estimation
        if $\lambda_i < \lambda_h$ then
          handover cluster head role to camera $c_h$
      else
        wait random backoff time
        if auction initiate was received then
          send object information
        else
          change role to cluster head
        if handover is received then
          change role to cluster head
```

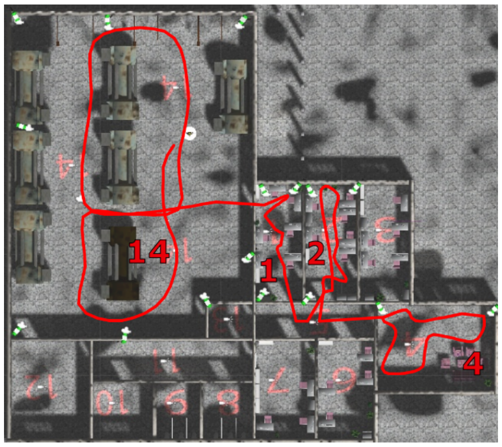
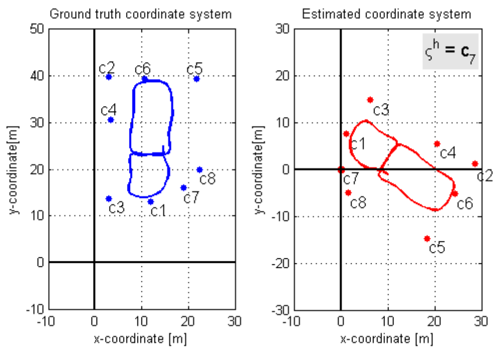
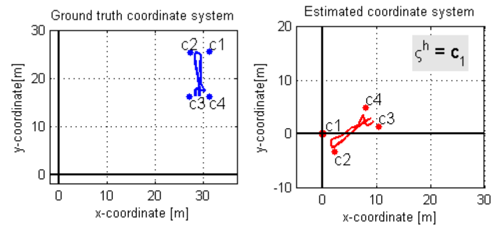


Fig. 3. Simulation scenario including 14 rooms, whereby the object was moving through five of them. The red path depicts the ground truth movement of the object.



(a) Room 14



(b) Room 2

Fig. 4. Results of the calibration and state estimation for (a) room 14 and (b) room 2 with a noise value of $k = 0.05$. The left plots show the ground truth coordinate systems and the right plots the self-calibrated coordinate systems. The estimated object paths match except to rotation and translation.

Cameras within the same room are connected, which enables them to establish a common coordinate system. This means, there are four different coordinate systems established by our algorithm, one for each room.

To enable an election of coordinate system and tracking cluster heads, we defined the utility as:

$$\lambda_i = \nu_i^j \cdot R_{total,i}$$

where ν_i^j is the detection confidence at camera c_i for object o_j and $R_{total,i}$ comprises the processing power, the remaining energy and the remaining memory at camera c_i.

To simulate sensor errors, we superimpose distance measurements to objects with a uniformly distributed error within the limits $[-\delta, +\delta]$. The limits increase with increasing distance to objects, since sensor measurements become typically less reliable the further the objects are away [2]. δ is thus modeled as a function of the distance and can be manipulated using the factor k:

$$\delta = k * d \tag{1}$$

Results: Figure 4 shows the comparison of the cooperative state estimation within the ground truth coordinate system (blue) and the self-calibrated coordinate system (red) for two of the five rooms. The ground truth coordinate systems are depicted in blue in the left plots and the self-calibrated coordinate systems in red in the right plots. This means, for the results in the left plots, cameras had ground truth information of all camera positions to perform the state estimation. The results in the right plots were generated by the cameras without having any a-priori topology or calibration information. Room 14, shown in figure 4 (a), is equipped with 8 cameras, while room 2, shown in figure 4 (b), is equipped with 4 cameras.

During the self-calibration, cameras c_7 and c_1 were elected as coordinate system heads c^h for room 14 and room 2, respectively. These two cameras are thus located in the origin of the self-calibrated coordinate systems. The resulting network calibration and object path estimations match the ground truths. However, they are shifted and rotated, since they were established from the viewpoints of the coordinate system heads.

Figure 5 shows the average state estimation error for the object paths per room over increasing noise. The error is expressed using the root-mean-square error (RMSE) defined as shown in equation 2:

$$RMSE = \sqrt{\frac{\sum_{j=1}^{s}(x_j^{true} - x_j^{est})^2 + (y_j^{true} - y_j^{est})^2}{s}} \tag{2}$$

where s is the number of position estimations, (x_j^{true}, y_j^{true}) is the ground truth position of object o_j and (x_j^{est}, y_j^{est}) is the estimated position. For noise values of up to 0.4, the average state estimation error never exceeds $1.6m$. Due to the self-calibration, camera positions are also affected by noise, which leads to more error prone state estimation results compared to approaches with an a-priori known ground plane.

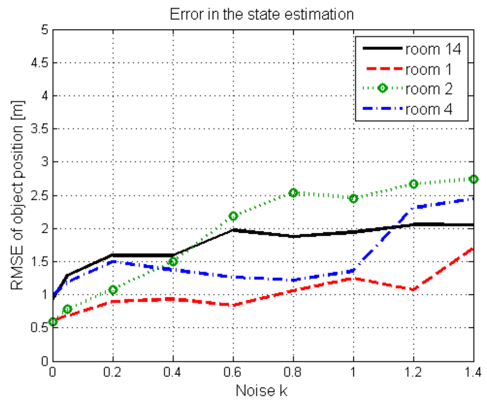


Fig. 5. RMSE of the object position after the self-calibration and state estimation.

Figure 6 shows the average number of sent messages per camera per room, which is required to calibrate the network. For a noise value of 0.4, cameras require on average between 30 and 40 message to calibrate the network. The communication costs are low but depend on the number of neighboring cameras and on the quality of the object observations. Having measurements with a reasonable error allows cameras to calibrate faster and more accurate. The larger the error, the more object observations, and therefore messages, are required. Detailed resource measurements of the state estimation itself can be found in [1].

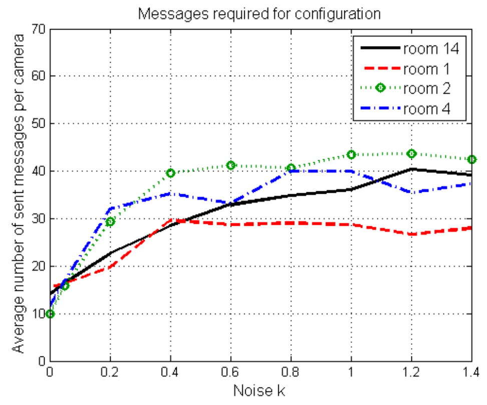


Fig. 6. Average number of sent messages required per camera to calibrate the network.

VI. Conclusion

In this paper we presented an algorithm which enables camera networks to calibrate themselves and then use this calibration information to perform collaborative object tracking. We thus handle errors in camera positions, established during the calibration process, as well as errors in object observations, obtained during the sensor measurements.

In our next steps, we will introduce a feed-back loop, which should improve the accuracy of the network calibration. For that purpose, state estimations of objects will be fed back into the calibration algorithm. Further, we will improve the selection of tracking cluster members, such that no longer all cameras which can see the object are selected as cluster members, but only a sufficient subset.

Acknowledgments

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] M. Schranz and B. Rinner, "Resource-aware dynamic clustering utilizing state estimation in visual sensor networks," *Sensors & Transducers*, vol. 191, no. 8, p. 28, 2015.

[2] J. Simonjan and B. Rinner, "Distributed visual sensor network calibration based on joint object detections," in *Proceedings of the International Conference on Distributed Computing in Sensor Systems, to appear*, 2017.

[3] C. Soto, B. Song, and A. K. Roy-Chowdhury, "Distributed multi-target tracking in a self-configuring camera network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 1486–1493.

[4] L. Liu, X. Zhang, and H. Ma, "Dynamic node collaboration for mobile target tracking in wireless camera sensor networks," in *Proceedings of the IEEE International Conference on Computer Communications*. IEEE, 2009, pp. 1188–1196.

[5] J. C. SanMiguel and A. Cavallaro, "Cost-aware coalitions for collaborative tracking in resource-constrained camera networks," *IEEE Sensors Journal*, vol. 15, no. 5, pp. 2657–2668, 2015.

[6] A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury, "Information weighted consensus filters and their application in distributed camera networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 12, pp. 3112–3125, 2013.

[7] 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.

[8] 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)*. IEEE, 2013, pp. 433–440.

[9] 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.

[10] A. I. Mourikis and S. I. Roumeliotis, "Predicting the performance of cooperative simultaneous localization and mapping (c-slam)," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1273–1286, 2006.

[11] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller, "Multiple relative pose graphs for robust cooperative mapping," in *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 3185–3192.

[12] C. Ding, B. Song, A. Morye, J. A. Farrell, and A. K. Roy-Chowdhury, "Collaborative sensing in a distributed ptz camera network," *IEEE Transactions on Image Processing*, vol. 21, no. 7, pp. 3282–3295, 2012.

[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] M. Schranz and B. Rinner, "Demo: Vsnsim - a simulator for control and coordination in visual sensor networks," in *Proceedings of the International Conference on Distributed Smart Cameras*. ACM, 2014, pp. 44:1–44:3.