Resilient Self-Calibration in Distributed Visual Sensor Networks

Jennifer Simonjan*, Bernhard Dieber†, Bernhard Rinner*
*Alpen-Adria-Universität Klagenfurt, Austria, Email: {Jennifer.Simonjan@aau.at}{Bernhard.Rinner@aau.at}
†JOANNEUM RESEARCH, Austria, Email: {Bernhard.Dieber@joanneum.at}

***Abstract*—Today, camera networks are pervasively used in smart environments such as intelligent homes, industrial automation or surveillance. These applications often require cameras to be aware of their spatial neighbors or even to operate on a common ground plane. A major concern in the use of sensor networks in general is their robustness and reliability even in the presence of attackers.**

This paper addresses the challenge of detecting malicious nodes during the calibration phase of camera networks. Such a resilient calibration enables robust and reliable localization results and the elimination of attackers right after the network deployment. Specifically, we consider the problem of identifying subverted nodes which manipulate calibration data and can not be detected by standard cryptographic methods. The experiments in our network show that our self-calibration algorithm enables location-unknown cameras to successfully detect malicious nodes while autonomously calibrating the network.

I. Introduction

Visual sensor networks (VSNs) consist of spatially distributed and interconnected smart cameras, which are not only capable of capturing images, but can also perform processing and communication. They are nowadays widely used in application areas such as surveillance, smart environments and industrial automation and have often strong requirements concerning resilience and reliability [1]. Furthermore, the majority of these applications require the cameras to be aware of either their spatial neighbors or the poses of all camera nodes [2], [3], [4]. Various network calibration algorithms have thus been introduced in the past years. However, they typically do not consider adversarial behavior of malicious nodes.

In this paper, we address a security problem which can not be solved by standard cryptographic methods. The two reasons for this problem are that 1) cryptography is too expensive for many embedded and resource-constrained sensor platforms and 2) subverted nodes that have been successfully authenticated before and have then been overtaken by attackers would not be detected. The latter is known as insider attack, performed by infiltrated nodes to actively disrupt the proper functioning of the network [5]. In the specific case of camera network calibration, malicious nodes may inject manipulated information into the network in order to falsify the calibration results to make others believe that they are positioned somewhere else, covering a certain area which is not covered in reality (and vice versa).

To address this problem, we present a fully distributed method which enables cameras to build up trust to others and thus to detect malicious nodes throughout the calibration phase of the network. To do so, cameras rely on a trust model which incorporates locally evaluated and remotely recommended trust. We demonstrate its functionality along with our self-calibration algorithm proposed in [6], [7].

Summarized, our main contribution is a resilient self-calibration algorithm for VSNs, which is able to detect malicious nodes while autonomously calibrating the network. The method is not bound to our self-calibration algorithm but can be used for any state-of-the-art camera network calibration algorithm. To the best of our knowledge, existing camera network calibration methods do not take malicious behavior into account.

The remainder of the paper is organized as follows. Section II discusses related work and Section III gives an overview of our VSN self-calibration algorithm. Section IV discusses the considered attack scenario and Section V introduces our proposed countermeasure. Evaluation results are presented in Section VI and Section VII concludes and outlines future work.

II. Related work

Sensor network calibration is a key enabling technique for many applications such as the Internet of Things and has been extensively studied within the past decades [8], [9], [10], [11]. Among various calibration solutions for different sensor network types, signal measurement based calibration is the most popular approach in wireless sensor networks (WSNs), while computer vision based calibration is most common in VSNs. An overview of the state-of-the-art in sensor network calibration is given in Table I. We categorized the networks into two major types, namely WSNs and VSNs, and included information about the resilience of the calibration technique.

Distributed sensor nodes have neither any a-priori knowledge of others, nor can they trust any other node in the network. Thus, achieving trust and reliability in sensor networks is typically a difficult task [12]. Traditional security solutions such as cryptography do not solve the problem of eliminating malicious behavior in a distributed network. Thus, novel solutions have to be developed in order to secure privacy sensitive networks such as VSNs.

Developing smart VSNs is a multi-disciplinary field related to computer vision, signal processing, communication, embedded computing and distributed systems. Most of the work for VSN calibration originates from the computer vision community, improving feature detection and matching across

various camera views in order to derive relative poses of the cameras. One example for a vision-based calibration was introduced by Devarajan et al. [8]. In their distributed algorithm, cameras detect feature points of the scene and match them with those of their overlapping neighbors to reason about relative poses. They also explored the advantages of a decentralized solution over a centralized one in terms of communication costs and processing power but did not consider any failures or adversarial behavior.

Two further vision-based calibration techniques were presented by Funiak et al. [13] and Ly et al. [14]. In the first approach, cameras collaboratively track a moving object to reason about consistent camera poses. Similar to our self-calibration algorithm proposed in [7], their algorithm provides pose estimates along with certainties. The second approach enables self-calibration for heterogeneous camera networks based on structure from motion. However, none of the approaches consider node failures or adversarial behavior.

Recently, Guan et al. [9] proposed an approach for VSN calibration, which is based on analyzing tracks of pedestrians. The 3D positions of head and foot points of the pedestrians are detected and used to determine the relative poses of the cameras. The authors claim that their approach results in simpler computations and a more flexible and accurate calibration compared to other methods. This solution is most similar to ours, also aiming at resource efficiency. However, failing or malicious nodes are not considered.

Santo et al. [11] presented a device-free, privacy preserving indoor calibration method using infrared cameras and retro-reflectors. The cameras capture retro-reflections from markers attached to walls and can thus detect persons by observing occlusions of markers. Therefore, persons are tracked while the camera-marker network is calibrated.

The state-of-the-art in VSN self-calibration showed, that the algorithms do typically not consider malicious or failing nodes. Robust calibration algorithms, taking adversaries into account, are usually proposed by the WSN community. One example approach, which is based on trust evaluation to overcome various attacks such as spoofing was introduced by Li et al. [15]. The trust evaluation is obtained via several calibration related properties including estimated distance, calibration performance and position information of beacon nodes.

Two recent approaches were presented by Yuan et al. [16] and Shi et al. [17]. The first one introduces a distributed calibration scheme for WSNs, which is based on anchor nodes and Received Signal Strength (RSS) measurements and is able to overcome sybil attacks. The second approach presents a distributed algorithm which detects malicious nodes throughout the calibration of the network based on geometric reasoning. The basic idea of this approach is similar to ours, exploiting plausibility checks for node positions.

Jiang et al. [18] proposed the Efficient Distributed Trust Model (EDTM) for WSNs. This trust model relies, unlike many others, not only on communication behavior but also on energy and data trust and enables to evaluate the trustworthiness of sensor nodes in general.

TABLE I
COMPARISON OF RESILIENT SENSOR NETWORK CALIBRATION METHODS

Reference	Objective	Method	Type	Malicious nodes
Devarajan et al. [8]	network calibration	scene point detection and matching	VSN	not considered
Funiak et al. [13]	network calibration	collaborative tracking	VSN	not considered
Ly et al. [14]	network calibration	structure from motion	VSN	not considered
Guan et al. [9]	network calibration	analyzing tracks of pedestrians	VSN	not considered
Santo et al. [11]	person tracking & network calibration	capturing retro-reflections from markers	VSN	not considered
Li et al. [15]	resilient network calibration	trust evaluation based on calibration results	WSN	considered
Yuan et al. [16]	resilient network calibration	based on anchor nodes and signal measurements	WSN	considered
Shi et al. [17]	resilient network calibration	plausibility checks based on geometric reasoning	WSN	considered
Jiang et al. [18]	distributed trust model	trust model based on local and recommended trust	WSN	considered
Our approach	resilient network calibration	trust model based on local and recommended trust	VSN	considered

Summarized, resilient and reliable calibration algorithms or trust evaluation models are typically developed for WSNs rather than VSNs, which led us to take our ideas from WSNs. However, a majority of them relies on distance estimations between nodes based on signal measurements and/or anchor nodes with known positions. Our VSN self-calibration approach does neither rely on signal measurements nor on anchor nodes. Further, cameras can typically exploit more information from their environment than scalar sensors, which is an advantage for security and reliability purposes.

III. VSN SELF-CALIBRATION

Our distributed and resource-aware self-calibration algorithm for VSNs was introduced in [6], [7]. It is a range-based localization algorithm, which enables networked cameras to determine relative poses of all others based on simple geometric computations and multi-lateration. Further, it enables the cameras to agree on one common coordinate system, which can then be used for applications running on top. To calibrate the network, cameras rely on the identification of shared objects and local distance and angle measurements. The main advantages of our approach over others are summarized as follows: 1.) it does not rely on any a-priori topological information or user-interaction, 2.) it exploits algorithms of low complexity and reduced communication and 3.) it is scalable.

A. Self-calibration algorithm

The self-calibration algorithm consists of three parts: 1) distance-angle estimation, 2) 1-hop neighbor calibration and 3) network-wide calibration. A high-level overview of the self-calibration procedure is given in Algorithm 1. In the distance-angle estimation, cameras construct the so-called observation vector υ, which includes information about locally detected objects. Observation vectors include local distance d and angle α measurements to the objects, a timestamp ts and an

object identifier f, and are broadcasted upon construction. The timestamp and the object identifier enable others to determine if they have observed the same object at the same time, while the distance and angle measurements enable them to geometrically reason about the pose of the camera. Geometric multi-lateration is exactly what happens in the 1-hop neighbor calibration, which calibrates cameras with overlapping FOVs (considered as neighbors) in a pairwise manner. A pairwise calibration performed by a camera c_h to localize another camera c_i results in a localization vector λ_{hi}, which includes the position (x_{hi}, y_{hi}) and the orientation φ_{hi} of camera c_i wrt. the local coordinate system of camera c_h. The localization vector λ_{hi} is used to spread the calibration information through the network in order to extend the calibration to multi-hop neighbors (network-wide calibration).

Along with spreading observation and localization vectors, cameras also elect a leader, whose local coordinate system is used as common one. This is required, since we do not assume any access to real-world coordinates but still aim to establish a single common coordinate system in the network. Summarized, the self-calibration algorithm generates a single common coordinate system with the poses of all camera nodes for connected networks. A detailed description and evaluation can be found in [6], [7].

Algorithm 1 Overview of the self-calibration algorithm

1. Distance-angle estimation

On object o_j is detected at camera c_i
  broadcast object observation vector υ_i including local distance d_{ij}, angle α_{ij}, timestamp ts and identifier f_{ij}

2. 1-hop neighbor calibration

On υ_i is received at camera c_h
  perform pairwise calibration based on multi-lateration
  broadcast localization vector λ_{hi} including $(x_{hi}, y_{hi}, \varphi_{hi})$

3. Network-wide calibration

On λ_{hi} is received at camera c_g
  if the sender c_h has already been localized **then**
    perform network-wide calibration
    broadcast localization vector λ_{gi} including $(x_{gi}, y_{gi}, \varphi_{gi})$

IV. Attack Scenario

Since cameras have access to highly sensitive data, VSNs have usually very high privacy and security concerns, and thus, malicious nodes and manipulated data need to be detected already in the calibration phase, right after the network deployment.

Most threat models for sensor networks categorize threats into two main classes, namely insider and outsider attacks [5], [19]. Outsider attacks are performed by non-authorized entities. In the case of a wireless sensor network an outsider may simply eavesdrop or jam the communication. Physical destruction or displacement of nodes are also considered as outsider attacks, whereby a sensor network is not able to distinguish between destroyed nodes or benign nodes that failed due to low battery, network disconnection or blocked views. Many outsider attacks such as eavesdropping or spoofing can easily be addressed using state-of-the-art authentication and encryption schemes. However, the cases of displaced or destroyed/failing nodes need to be considered and handled by calibration algorithms in order to not provide false calibration results.

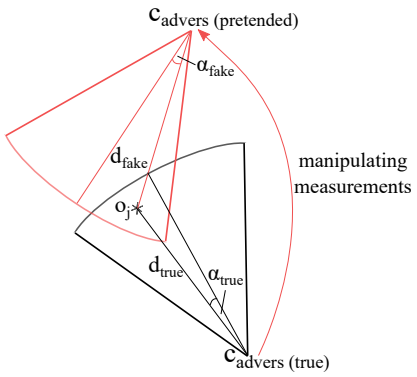


Fig. 1. Data manipulation attack. To manipulate its position, an adversary c_{advers} simply pretends to observe objects from a fake distance d_{fake} and angle α_{fake}, rather than from the true ones d_{true} and α_{true}.

Insider attacks enable adversaries to act as legitimate nodes and thus to actively disrupt the sensor network by manipulating data. Such an attack may be launched by subverted nodes (i.e., formally legitimate participants of the sensor network that have been taken over by an attacker), which already completed the authentication process successfully. In the specific case of network calibration, subverted malicious nodes may falsify data on purpose in order to manipulate the calibration results. Specifically, they might pretend to be positioned somewhere else to make others believe that a certain sensitive area (e.g., cash machine) is covered, while it is not in reality. Tampering a pose is typically easy in an autonomous, distributed network in which sensor nodes can solely rely on local and received information. Malicious nodes can thus simply pretend to be located somewhere else by injecting manipulated data into the network.

In the specific case of our self-calibration, adversaries would broadcast fake object observation vectors υ. Figure 1 shows the data manipulation attack, which is achieved by manipulating the pose of the camera and thus the local distances and angles included in the object observation vectors. Instead of broadcasting the true measurements, the adversary broadcasts tampered measurements d_{fake} and α_{fake}, which leads others to locate the node with a false pose.

V. Robust Self-Calibration Algorithm

Adversary detection in a fully distributed and autonomous network is non-trivial since nodes do not have any previous knowledge about others and can solely rely on sensed or received information. To increase the robustness of the

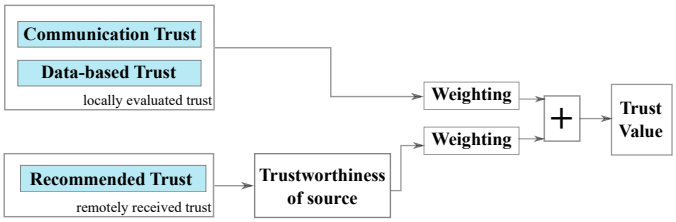


Fig. 2. Trust model diagram adopted from [18].

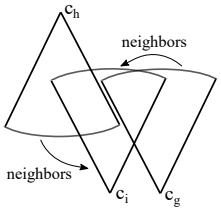


Fig. 3. Example camera network. Network of three cameras c_i, c_h and c_g, whereby only camera c_i is overlapping with both of the other cameras.

self-calibration, cameras need to cope with failures and to detect malicious nodes to eliminate manipulated data. We will see, that the distributed and flexible nature of our self-calibration algorithm is able to inherently handle the cases of displaced and destroyed or failing nodes. For the purpose of adversary detection, cameras construct and maintain a trust graph throughout the self-calibration, which includes trust probabilities to all other nodes.

A. Outsider attacks: destruction and displacement

If an attacker physically destroys a node or if a benign node fails during the self-calibration process, it won't be included in the resulting calibrated network. This merely affects the rest of the network, if the node was the only connection between two network regions. Our self-calibration algorithm is designed in a way, that cameras of connected networks elect a network leader, whose local coordinate system is used as common ground plane. If a network is split into two disconnected regions, the nodes of each region elect their own network leader, resulting in two ground planes. Thus, the self-calibration still works, but results in two common ground planes, one for each network region.

If destruction or failing happens after a successful self-calibration, other nodes recognize it due to the missing communication. For calibration purposes, cameras exchange object observation vectors. However, to include topology changes, cameras keep on broadcasting these vectors also after the calibration finished. Whenever a camera realizes that there were no observations received from a neighbor for a certain amount of time, it requests a *Keep Alive* message from the respective camera. If no reply is received, the camera is marked as *Down* and all other network nodes are informed about the failure of the camera. The camera is simply excluded from the network, which does not affect the common ground plane or network stability. The mechanism of broadcasting observation vectors even after a successful calibration also ensures the detection and re-calibration of displaced nodes.

B. Insider attacks: data manipulation

In order to eliminate malicious nodes, cameras construct and maintain a trust graph, which defines the likelihood of being an adversary for all other known nodes in the network. The trust model used to establish trust graphs is adopted from Jiang et al. [18] and is shown in Figure 2. Based on this model, trust graphs are built throughout the self-calibration procedure. As the model shows, the trust is built upon locally evaluated and remotely received trust. Locally evaluated trust includes communication and data-based trust and remotely received trust includes recommended trust. The data-based trust proposed in [18] relies on the assumption of spatial correlation between data packets, which means that packets sent among neighboring nodes are similar. This does not hold true for visual sensor networks for one major reason: being directional sensors, cameras may observe scenes from very different viewpoints even if they are neighbors. Thus, data-based trust for VSNs needs to be modeled differently. In order to ease the explanations in the following subsections, we assume a camera network of three cameras, namely c_i, c_h and c_g, arranged as shown in Figure 3.

1) Communication trust: Communication trust is evaluated locally on the camera nodes based on the communication behavior of another camera, which can be both an overlapping camera (neighbor) or a non-overlapping camera. If proportionally large numbers of messages or messages with meaningless content are received from any camera c_h at camera c_i, the trust value τ_{ih} of camera c_h is decreased by a value δ.

2) Data-based trust: Data-based trust relies on plausibility checks based on geometric reasoning and is evaluated by cameras for neighbors only. An overview of the data-based trust algorithm is shown in Algorithm 2.

Plausibility checks are triggered by receiving object observation vectors from neighboring cameras. If a camera c_i receives an observation vector υ_{hj} from camera c_h for object o_j, it first checks the plausibility of the distance d_{hj} and angle α_{hj} measurements included in the vector. In case these measurement values are too small or too large to be plausible, the trust τ_{ih} of camera c_h is decreased by a value δ. Further, if a large amount of received observation vectors $\upsilon_{hj}, ..., \upsilon_{hn}$ still lead to a non-solvable multi-lateration problem at camera c_i, the neighbor might have sent arbitrary object observations, and thus its trust value τ_{ih} is decreased by δ.

Once a neighbor c_h was successfully localized, the localizing camera c_i is aware of the claimed pose of the neighbor and can thus check its plausibility. To do so, camera c_i first evaluates the overlap with camera c_h based on the indicator function shown in Equation 1.

$$overlap(c_i, c_h) = \begin{cases} 1, & F_i \cap F_h \neq 0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where F_i and F_h are the areas covered by the respective camera. If $overlap(c_i, c_h) = 0$, the trust value τ_{ih} is decreased by δ.

Further, for every detected object o_j, camera c_i is able to determine if also neighbor c_h can see the object. This plausibility check is expressed by the following function:

$$cover(c_h, o_j) = \begin{cases} 1, & \text{if } o_j \text{ is in the FOV of } c_h \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

If $cover(c_h, o_j) = 1$, camera c_i sends an object request req_j to the neighbor c_h. This request simply includes the object identifier f_{ij}, requesting distance d_{hj} and angle α_{hj} measurements from camera c_h to this specific object. If neighbor c_h is not able to provide an appropriate answer including plausible measurements or an answer at all, its trust value τ_{ih} is decreased by δ. If a reply rep_j is received, camera c_i compares the received object measurements o_{hj} with the theoretically calculated measurements o^i_{hj} to determine plausibility. This plausibility check is indicated by the following function:

$$equals(o_{hj}, o^i_{hj}) = \begin{cases} 1, & \text{if } (d_{hj}, \alpha_{hj}) = (d^i_{hj}, \alpha^i_{hj}) \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where (d_{hj}, α_{hj}) are the object measurements received from camera c_h and $(d^i_{hj}, \alpha^i_{hj})$ are the theoretically calculated ones. If $equals(o_{hj}, o^i_{hj}) = 1$, the trust value of the neighbor is increased by δ, otherwise it is decreased.

Algorithm 2 Data-based trust

On υ_{hj} from camera c_h is received at camera c_i
  if $cover(c_i, o_j) = 1$ **then**
    if (d_{hj}, α_{hj}) is not plausible **then**
      decrease trust value τ_{ih} by δ
    else
      localize camera c_h through multi-lateration
      if non-solvable multi-lateration **then**
        decrease trust value τ_{ih} by δ
      else
        if $overlap(c_i, c_h) = 0$ **then**
          decrease trust value τ_{ih} by δ

On object o_j is detected at camera c_i
  if camera c_h has already been localized **then**
    if $cover(c_h, o_j) = 1$ **then**
      send object request req_j to camera c_h
      theoretically determine $(d^i_{hj}, \alpha^i_{hj})$
    if rep_j from camera c_h is received at camera c_i **then**
      if $equals(o_{hj}, o^i_{hj}) = 1$ **then**
        increase trust value τ_{ih} by δ
      else
        decrease trust value τ_{ih} by δ
    else
      decrease trust value τ_{ih} by δ

3) Recommended trust: Cameras also take remotely received trust values into account if the recommending neighbor was already proven to be trustworthy. This means, that if a trustworthy neighbor c_i informs a camera c_g about a malicious or a trustworthy third node c_h, camera c_g will consider this suggestion. An overview of the recommended trust algorithm is shown in Algorithm 3.

In general, there are two different cases: Node c_h, included in the trust suggestion, is a neighbor itself of camera c_g, or is non-overlapping and thus a multi-hop neighbor of camera c_g. In the first case, camera c_g is also able to evaluate the data-based trust of camera c_h and accumulates it with the recommended trust from camera c_i (see Figure 8). In the second case, camera c_g can accumulate recommended trust with communication trust only, since it is not overlapping with camera c_h and thus not able to evaluate any data-based trust.

4) Trust determination: Two thresholds t_{low} and t_{high} are used to determine whether a camera is trustworthy or not. If the trust value τ_{ih} of camera c_h falls below the threshold t_{low}, the camera is marked as malicious. If the trust value τ_{ih} on the other hand exceeds the threshold t_{high}, camera c_h is marked as trustworthy. In both cases, camera c_i broadcasts a trust message Υ_{ih} to inform all other cameras about the detected trustworthiness of the camera. Trust values which did not exceed any of the thresholds indicate that there was no trust decision made yet, and the node is neither considered to be malicious nor to be trustworthy. The thresholds t_{low} and t_{high} need to be determined experimentally, however, the smaller t_{low} and the larger t_{high}, the higher the robustness but the longer the duration to evaluate trustworthiness. Algorithm 4 shows an overview of the trust determination.

Algorithm 3 Recommended trust

if trust message Υ_{ih} from camera c_i is received at c_g **then**
  if camera c_i is trustworthy **then**
    if camera c_h is a neighbor of camera c_g **then**
      accumulate recommended, communication and data-based trust to form τ_{gh}
    else
      accumulate recommended and communication trust to form τ_{gh}
  else
    drop trust message

Algorithm 4 Trust determination

if $\tau_{ih} < t_{low}$ **then**
  mark camera c_h as malicious
  broadcast trust message Υ_{ih} including trust value τ_{ih}
else if $\tau_{ih} > t_{high}$ **then**
  mark camera c_h as trustworthy
  broadcast trust message Υ_{ih} including trust value τ_{ih}

VI. Evaluation

This section discusses the evaluation results we achieved in our experiments. The evaluation in this paper concentrates on evaluating the robustness of the algorithm, however, the results imply a working self-calibration. Detailed evaluation results of the self-calibration algorithm in terms of performance, accuracy and communication costs can be found in [6], [7].

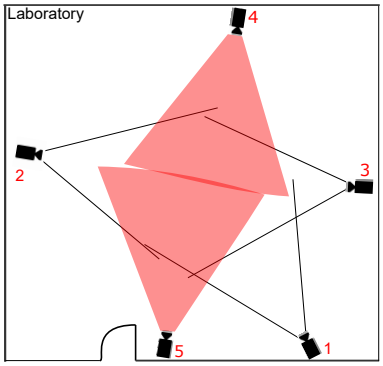


Fig. 4. The deployment setup of our network consisting of five smart cameras and their respective FOVs. Red colored FOVs depict malicious nodes.

A. Camera network

To evaluate our algorithm under realistic conditions, we deployed it in a camera network consisting of five distributed smart cameras connected via Ethernet cables. Each camera is composed of a simple web-camera attached to an Intel NUC[1] computer and runs the resilient self-calibration algorithm. As a middleware, the Robot Operating System[2] (ROS) [20] is used, which is an environment for topic-based publish/subscribe. ROS thus takes care about distributed messaging without requiring the senders and receivers to be aware of each other.

B. Evaluation scenario

Figure 4 shows the network setup, including all cameras and their FOVs, in our laboratory. The triangles depict the FOVs of the cameras, whereby red colored FOVs indicate malicious cameras. The network consists of 5 cameras, from which we manipulated two (camera 4 and 5) to be malicious. Malicious nodes forge their positions, resulting in fake object observation vectors, in order to pretend to be located somewhere else. To generate different fake poses for different adversaries, we created fake positions (x,y) and orientations ϕ as follows:

$$x_{fake} = x_{true} + a_s \cdot rand_x \tag{4}$$

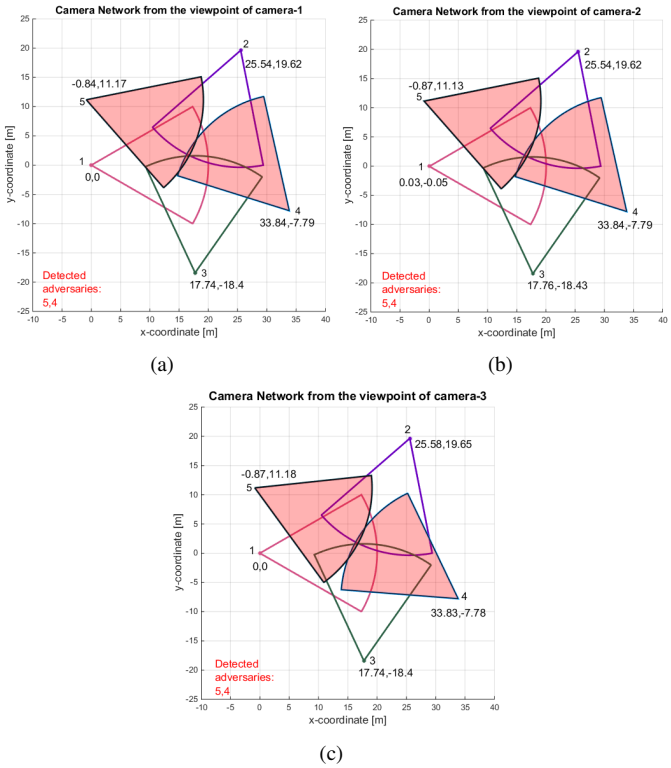
$$y_{fake} = y_{true} + a_s \cdot rand_y \tag{5}$$

$$\phi_{fake} = \phi_{true} + a_s \cdot rand_\phi \tag{6}$$

where x_{true} and y_{true} are the true coordinates of the camera, a_s is the attack severeness and $rand_x$ and $rand_y$ are either 1 or -1 with a probability of 50% each. Thus, x_{fake} and y_{fake} can be positive or negative. Fake camera orientations ϕ_{fake} are generated by adding/subtracting $a_s \cdot rand_\phi$ to/from the true orientation. The variable $rand_\phi$ is either 1 or -1, with a probability of 50% each. Further, we defined the threshold values for trust decisions as $t_{low} = -5$ and $t_{high} = 5$, set initial trust values τ to 0 and in-/decreased trust values by $\delta = 1$.

Since our evaluation aims at assessing the performance of the malicious node detection, we use synthesized object detection data in order to exclude potential errors in sensor measurements from our results. The synthesized data models twice as many objects as cameras, which moved randomly through the network, eventually leaving the area of interest. We deployed the synthesized data together with the algorithm to each camera.

[1]https://www.intel.de/content/www/de/de/products/boards-kits/nuc.html
[2]http://www.ros.org/



(a) (b) (c)

Fig. 5. Resilient self-calibration results in form of the common ground planes which were estimated locally by each camera. The three plots (a), (b) and (c) show the local estimations of camera 1,2 and 3, respectively, for an attack severeness of 3. Malicious nodes are depicted by red FOVs and their IDs in the lower left corner.

C. Results

Figure 5 shows the resilient self-calibration results for the network from the viewpoint of the cameras 1(a), 2(b) and 3(c) for an attack severeness of 3. This means, the figures depict the locally estimated ground planes of all benign cameras. First of all, we can see that the cameras successfully agreed on a common ground plane, since the local views match each other. Cameras which were detected to be malicious are depicted by red FOVs and their IDs in the lower left corner of the plots. Compared to the ground truth network in Figure 4, we can see that the networks match except to a global rotation and translation and that the cameras were able to detect both adversaries, namely camera 4 and 5, correctly.

Figure 6 shows the assembly of the trust values for the other nodes in the network from the viewpoint of the benign cameras 1(a), 2(b) and 3(c). The node IDs are depicted on the x-axis and the trust values on the y-axis. Blue bars depict locally evaluated trust and green bars depict remotely received

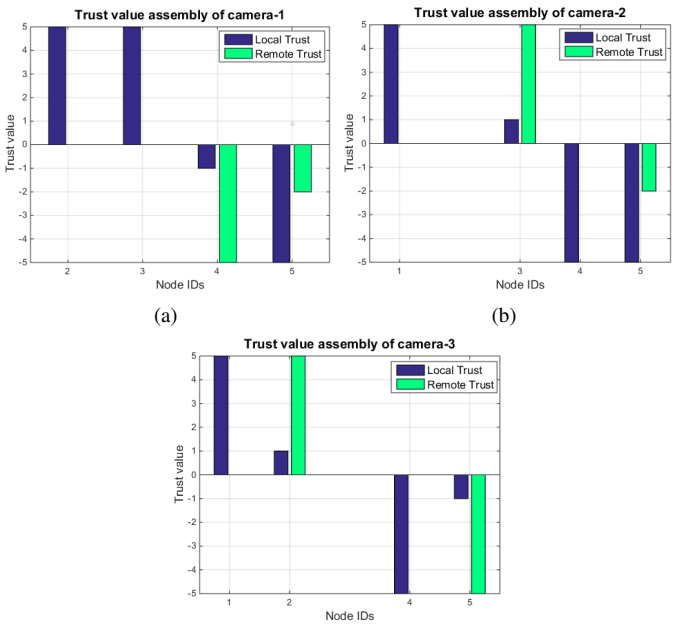


Fig. 6. The assembly of the trust values for the other nodes in the network from the viewpoint of the benign cameras 1(a), 2(b) and 3(c). Blue bars depict locally evaluated trust and green bars depict remotely received trust.

trust. As mentioned earlier, a trust decision is made whenever a trust value reaches a threshold value, thus the maxima of the bars are ± 5. Figure 6(a) shows for example, that camera 1 determined camera 2 and 3 to be trustworthy based on locally evaluated trust only. Camera 4 and 5 were found to be malicious, whereby the trust decision of camera 4 mainly relies on recommended trust. This is due to the fact that there is only a very small overlap between camera 1 and 4.

Figure 7 shows the average number of correct and false adversary detections per camera over an increasing attack severeness from 1 to 8. The influence of the attack severeness can be seen in Equations 4-6. The blue line depicts the true number of malicious nodes, the orange line the number of correctly detected adversaries and the yellow line the number of falsely detected adversaries. The algorithm ran 10 times per attack severeness and the results were averaged. As it can be seen from the plots, cameras were on average able to detect 1.8 adversaries out of 2 and the false detection rate was very low. Further, the number of correctly detected adversaries increases with the attack severeness, which is due to the fact that larger deviations between true and faked positions are easier to detect through plausibility checks.

Figure 8 shows four snapshots of the trust graph from the local viewpoint of camera 1. The snapshots were taken whenever a camera was localized or found to be trustworthy/malicious. The locally estimated camera network is depicted in gray. Depending on the current trust level, the thickness of the lines to other cameras varies. Thick lines indicate high trust to the other camera and thin lines indicate low trust. Figure 8(a) depicts the first snapshot of the trust graph. At this time, none

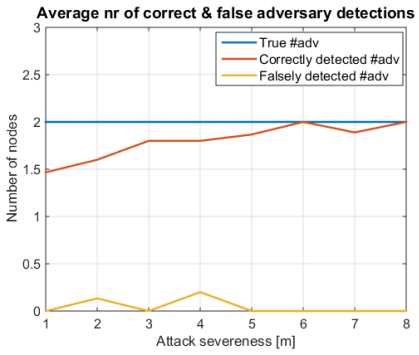


Fig. 7. Average number of correct and false adversary detections per camera over varying attack severeness. The blue line depicts the true number of malicious nodes, the orange line the number of correctly detected adversaries and the yellow line the number of falsely detected adversaries.

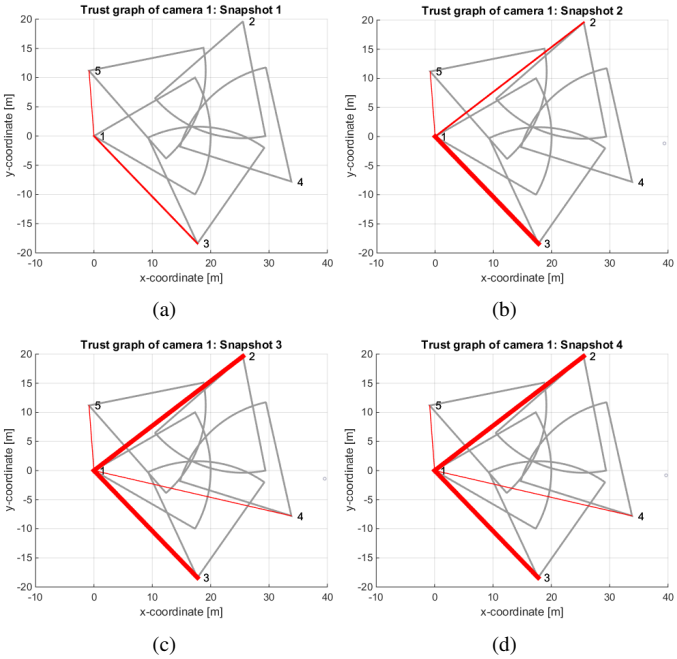


Fig. 8. Four snapshots of the local trust graph state at camera 1. Thick lines indicate high trust and thin lines indicate low trust.

of the other cameras is considered as trustworthy, since only two thin lines are connecting cameras 3 and 5. Whenever new trust values are gathered, the trust graph is updated. From snapshot 2 in Figure 8(b) we can see that the trust to camera 3 increased while also the trust evaluation for camera 2 started. Snapshot 3 in Figure 8(c) shows already a final state of the trust graph, in which cameras 2 and 3 are considered as trustworthy and cameras 4 and 5 as malicious. This result matches the ground truth shown in Figure 4.

Regarding the communication effort, we estimated the minimum number of additional messages required for building trust relations as follows:

$$msg_{trust} = (t_{high} \cdot c_{onehop} + 1) + c_{multihop}$$

where t_{high} is the upper threshold value for trust decisions, which we set to 5 in our experiments, c_{onehop} is the number of overlapping neighbors in the network, and $c_{multihop}$ is the number of non-overlapping cameras in the network. Cameras perform plausibility checks, including one message each, for overlapping neighbors until either the upper threshold t_{high} or the lower threshold t_{low} is exceeded. Upon exceeding the threshold, a trust decision is made and one message is sent out to inform others about the trustworthiness of the neighbor. In the case of non-overlapping cameras, a camera can only rely on received trust. Whenever a trust decision for a non-overlapping camera is made, one message is sent out to broadcast the information further. For our network and a trust threshold of $t_{high} = 5$, we estimated the minimum trust message count on the benign cameras to be 21.

Using rosbag[3], which is a tool for evaluating the communication in a distributed ROS environment, we recorded all published messages until camera 1 finished its trust graph and filtered out the trust related ones. Rosbag recorded 40 plausibility check requests req_j, 66 replies rep_j and 10 trust decision messages Υ. Requests and decisions are only sent out by benign nodes, whereas answers are provided by all nodes. The average number of trust related messages sent by benign cameras was thus

$$\frac{40}{3} + \frac{66}{5} + \frac{10}{3} \approx 30.$$

This very small overhead would increase only with increasing number of neighbors, but not with an increasing network.

VII. Conclusion

We presented a resilient self-calibration algorithm for camera networks, which enables cameras to estimate the poses of all other nodes in the network in a distributed, robust and autonomous way. To do so, cameras evaluate the trustworthiness of all other nodes, which is used to construct and maintain trust graphs. Trust graphs are built upon our trust model, which incorporates locally determined and remotely received trust. We conducted experiments in a network of five cameras to show the functionality of our algorithm. Future work will include further security extensions to cover a larger variety of attacks and an extension of our trust model to enable nodes considered as malicious to get back into a trustworthy state. Further, we will show that also attacks at the middleware layer can be compensated. Therefore, we will exploit the insecure architecture of ROS [21] to manipulate the network nodes and evaluate the impacts on the self-calibration. The major requirement of future sensor networks that operate in public will be reliability and robustness, in order to also provide privacy.

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 as well as by the Austrian Ministry for Transport, Innovation and Technology (bmvit) within the project framework CredRoS.

[3] http://wiki.ros.org/rosbag

References

[1] M. Reisslein, B. Rinner, and A. Roy-Chowdhury, "Smart camera networks [guest editors]," *Computer*, no. 5, pp. 23–25, 2014.

[2] 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.

[3] J. Simonjan, M. Schranz, and B. Rinner, "Self-calibration and cooperative state estimation in a resource-aware visual sensor network," in *Proceedings of the International Conference on Distributed Smart Cameras*. ACM, 2017.

[4] A. T. Kamal, J. H. Bappy, J. A. Farrell, and A. K. Roy-Chowdhury, "Distributed multi-target tracking and data association in vision networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 7, pp. 1397–1410, 2016.

[5] E. Shi and A. Perrig, "Designing secure sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 38–43, 2004.

[6] J. Simonjan and B. Rinner, "Decentralized and resource-efficient self-calibration of visual sensor networks," *Ad Hoc Networks*, vol. 88, pp. 112–128, 2019.

[7] ——, "Distributed visual sensor network calibration based on joint object detections," in *Proceedings of the 13th International Conference on Distributed Computing in Sensor Systems*, June 2017, pp. 109–116.

[8] D. Devarajan, R. J. Radke, and H. Chung, "Distributed metric calibration of ad hoc camera networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 3, pp. 380–403, 2006.

[9] 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.

[10] S.-Z. Wang, Y. Li, and W. Cheng, "Distributed classification of localization attacks in sensor networks using exchange-based feature extraction and classifier," *Journal of Sensors*, vol. 2016, 2016.

[11] H. Santo, T. Maekawa, and Y. Matsushita, "Device-free and privacy preserving indoor positioning using infrared retro-reflection imaging," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2017, pp. 141–152.

[12] L. Chen, S. Thombre, K. Jarvinen, E. S. Lohan, A. Alen-Savikko, H. Leppakoski, M. Z. H. Bhuiyan, S. Bu-Pasha, G. N. Ferrara, S. Honkala *et al.*, "Robustness, security and privacy in location-based services for future iot: A survey," *IEEE Access*, vol. 5, pp. 8956–8977, 2017.

[13] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, "Distributed localization of networked cameras," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*. ACM, 2006, pp. 34–42.

[14] D. S. Ly, C. Demonceaux, P. Vasseur, and C. Pégard, "Extrinsic calibration of heterogeneous cameras by line images," *Machine Vision and Applications*, vol. 25, no. 6, pp. 1601–1614, 2014.

[15] P. Li, X. Yu, H. Xu, J. Qian, L. Dong, and H. Nie, "Research on secure localization model based on trust valuation in wireless sensor networks," *Security and Communication Networks*, vol. 2017, 2017.

[16] Y. Yuan, L. Huo, Z. Wang, and D. Hogrefe, "Secure apit localization scheme against sybil attacks in distributed wireless sensor networks," *IEEE Access*, vol. 6, pp. 27 629–27 636, 2018.

[17] W. Shi, M. Barbeau, J.-P. Corriveau, J. Garcia-Alfaro, and M. Yao, "Secure localization in the presence of colluders in wsns," *Sensors*, vol. 17, no. 8, p. 1892, 2017.

[18] J. Jiang, G. Han, F. Wang, L. Shu, and M. Guizani, "An efficient distributed trust model for wireless sensor networks," *IEEE Transactions on Parallel & Distributed Systems*, vol. 26, no. 5, pp. 1228 – 1237, 2015.

[19] A. A. Cardenas, T. Roosta, and S. Sastry, "Rethinking security properties, threat models, and the design space in sensor networks: A case study in scada systems," *Ad Hoc Networks*, vol. 7, no. 8, pp. 1434–1447, 2009.

[20] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[21] B. Dieber, B. Breiling, S. Taurer, S. Kacianka, S. Rass, and P. Schartner, "Security for the robot operating system," *Robotics and Autonomous Systems*, vol. 98, pp. 192 – 203, 2017.