Machine Vision and Applications (2019) 30:671–688
https://doi.org/10.1007/s00138-019-01021-z

ORIGINAL PAPER

Auto-calibration of depth camera networks for people tracking

Otto Korkalo[1] · Tommi Tikkanen[1] · Paul Kemppi[1] · Petri Honkamaa[1]

Received: 8 April 2017 / Revised: 2 December 2018 / Accepted: 8 March 2019 / Published online: 28 March 2019
© The Author(s) 2019

Abstract
We address the problem of calibrating an embedded depth camera network designed for people tracking purposes. In our system, the nodes of the network are responsible for detecting the people moving in their view, and sending the observations to a centralized server for data fusion and tracking. We employ a plan-view approach where the depth camera views are transformed to top-view height maps where people are observed. As the server transforms the observations to a global plan-view coordinate system, accurate geometric calibration of the sensors has to be performed. Our main contribution is an auto-calibration method for such depth camera networks. In our approach, the sensor network topology and the initial 2D rigid transformations that map the observations to the global frame are determined using observations only. To distribute the errors in the initial calibration, the transformation parameters and the estimated positions of people are refined using a global optimization routine. To overcome inaccurate depth camera parameters, we re-calibrate the sensors using more flexible transformations, and experiment with similarity, affine, homography and thin-plate spline mappings. We evaluate the robustness, accuracy and precision of the approach using several real-life data sets, and compare the results to a checkerboard-based calibration method as well as to the ground truth trajectories collected with a mobile robot.

Keywords Depth camera network · Auto-calibration · Depth distortion · Thin-plate spline · People tracking

1 Introduction

Depth cameras have multiple advantages compared to conventional 2D cameras in people detection and tracking applications. As stated in [12], depth is a powerful cue for foreground segmentation, 3D shape and metric observations simplify foreground object classification, occlusions can be detected and handled more explicitly, and the third dimension can be used for the prediction step in tracking. Shape matching-based detection may take the advantage of 3D information when evaluating the fitness of the model (e.g. [30]), and gradient-based methods become more robust as the depth image gradients depend only on scene geometry and not on texture properties (e.g. [23,29]). Typically, depth cameras are based on active illumination, and thus, they are not as sensitive to changes in lighting conditions such as 2D cameras, and they can also operate in completely dark environments. An important advantage of using depth in people tracking is that depth-based systems are privacy preserving which can be a precondition for public installations.

Early depth-based people tracking systems used stereo [5] or time-of-flight (ToF) cameras [4] to generate disparity or depth maps from the scene. However, stereo imaging suffers partly from the same problems as 2D cameras, and the first ToF cameras were expensive and rather inaccurate for practical use in the people tracking context. Microsoft Kinect was the first commodity device with an accurate depth camera and an operation range suitable for people tracking applications. Since the Kinect device was released, several studies on depth-based and RGB-D (RGB and depth)-based people detection and tracking have been carried out (see [8] for a survey). Many works focus particularly on detection from the side view using template matching-based approaches (e.g. [15,23,29]). Such methods are beneficial for moving camera platforms such as mobile robots, but the drawback is that they are computationally intensive and cannot be efficiently used in embedded platforms. Additionally, they suffer from occlusions, and the camera position and orientation are restricted to certain views. Another common technique, especially for static camera installations, is the plan-view approach [12]. In the approach, the depth map is first transformed to a 3D point cloud in the camera coordinate system, which is rotated and

✉ Otto Korkalo
otto.korkalo@vtt.fi

[1] VTT Technical Research Centre of Finland, Vuorimiehentie 3, Espoo P.O. Box 1000, FI-02044 VTT, Finland

rendered to generate a top-view height map or occupancy map of the scene [5,12]. Assuming a static scene, people can be detected from the height or occupancy map, e.g. with local maxima search [31] or template matching techniques [26].

As depth cameras have limited range and field of view, multi-camera systems are used to enlarge the tracking volume. They also reduce occlusion-related problems in configurations where the cameras are installed at low angles and people occlude each other. In the first scenario, the cameras are typically installed so that their overlap is minimal or non-existent; and in the second scenario, the system takes the advantage of capturing the same scene from multiple views. Whether the goal is to fuse the original point clouds or preprocessed information such as people locations, the data must be aligned to a common coordinate system. Thus, the relative pose of the cameras and possibly the intrinsics (sensor and lens characteristics) has to be determined.

With 2D cameras, a common technique for multi-camera calibration is to use a known calibration target, such as a checkerboard plane, and capture multiple images of the target in different poses. The calibration routine estimates the parameters of the cameras by minimizing the total reprojection error of the detected checkerboard corners [32]. In [25], a laser pointer was used as a calibration target. Using 3D reconstruction techniques, the approach estimated the scene structure (laser pointer positions) and the camera parameters simultaneously. Similar techniques have also been proposed for RGB-D camera systems. For example, in [2,19], a number of RGB and depth frames from a checkerboard plane were collected to solve the topology of the camera network and the initial poses of the cameras. Then, a global bundle adjustment routine minimizing the total reprojection error of the checkerboard corners and the corresponding errors in depth was used to refine the parameters. In [21], spherical calibration targets were used to calibrate the extrinsics (position and orientation) of multiple depth cameras. A globally optimal solution for both the centres of the spheres as well as for the camera parameters was found by minimizing the reprojection error of the spheres in depth images.

Manual calibration of a camera network is a tedious task, and in practical applications, auto-calibration methods are preferred. The goal of camera network auto-calibration is to automatically solve the network topology, the extrinsics and possibly the intrinsics of the cameras so that they can operate in a common coordinate system. It is a widely studied problem with 2D cameras (e.g. [28] reviews such methods), but only a few works address the problem with depth camera networks. The method presented in [27] employs the plan-view approach for depth-based people tracking. In the approach, the cameras are aligned to a common top-view coordinate system using pairwise affine transformations, and one large height map of the scene is constructed. The transformations are defined by matching the simultaneous observations of people, but it remains unclear how they are estimated in a multi-camera scenario. In [18], a method for registering two depth cameras without user input is presented. In the approach, the moving objects (people) are extracted from each camera with a depth histogram-based background subtraction, and the corresponding point cloud centroids are used with RANSAC (random sample consensus) to search for a rough estimate of the rigid transformation between the cameras. The result is refined with a grid-search-based optimization routine that optimizes transformation parameters as well as temporal differences (time synchronization) between the cameras. In the previous approaches, the cameras' intrinsics are calibrated beforehand.

In this paper, we address the problem of calibrating depth camera networks such that the observations of different sensors are accurately aligned for data fusion and tracking. Our main contribution is an auto-calibration method for people tracking systems that utilize the plan-view approach. With the proposed method, the sensor network topology and the 2D transformations that map the sensor observations to the common coordinate system are solved automatically using observations from people as the only input. We also propose using thin-plate-spline (TPS) mappings to compensate for linear and nonlinear measurement errors directly in plan-view domain. We evaluate how different mappings perform, and experiment with rigid, similarity, affine, homography and TPS mappings. For the evaluation, we recorded several real-life data sets from four different depth camera configurations consisting of three to six cameras. We conducted multiple experiments to analyse the robustness, accuracy and precision of the method and compared the results to a checkerboard-based calibration method and to the ground truth data collected with a mobile robot.

Our approach has many benefits. It enables fast deployment of depth camera-based people tracking systems without the need of accurate pre-calibration of the intrinsics or manual calibration of the extrinsics. As the system is calibrated from the observations made of real people, neither special calibration targets nor manoeuvres are needed. The only requirements are that sensors have at least partial pairwise overlap, and form a connected graph. The method does not make any assumptions about the systematic measurement errors of the sensors, and it can be used with any kind of depth imaging device.

2 Methods

2.1 System overview

Our people tracking system consists of multiple distributed depth camera nodes and a centralized server. The nodes are responsible for detecting people in their own plan-view coordinate system, whereas the server maps the observations to a global coordinate frame, performs data fusion and constructs the spatio-temporal tracks for each person visible in one or more cameras. We assume that the camera views are partly overlapping, and that they form a connected graph. The cameras can be installed at any angle or height.

We use low-cost ARM-based single-board PCs as embedded computing platforms, therefore lightweight depth processing methods are desirable. We employ the plan-view approach with background subtraction to detect people from the depth frames. In the approach, captured depth frames are first converted to 3D point clouds which are then rotated to top-view perspective. A height map image is created from the rotated point cloud, and the people are detected with local maxima search. The observations are transformed back to the plan-view domain (the 2D metric coordinate system of the floor plane), and sent to the server. We call the pose parameters (orientation, height) of the cameras *the local extrinsics*, and the mappings that transform the observations to a global frame as *the global extrinsics* of the sensors. The camera focal length, principal point, lens model parameters and depth distortion parameters are called *the intrinsic parameters* or *the intrinsics*. Figure 1 shows the different modules, transformations and relations of the system.

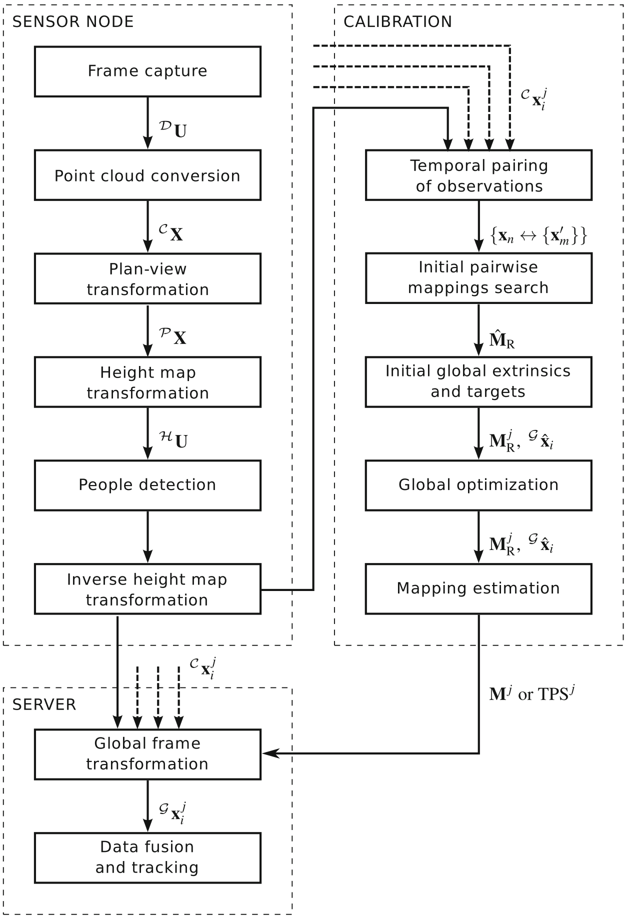
SENSOR NODE
Frame capture
$^{\mathcal{P}}\mathbf{U}$
Point cloud conversion
$^{\mathcal{C}}\mathbf{X}$
Plan-view transformation
$^{\mathcal{P}}\mathbf{X}$
Height map transformation
$^{\mathcal{H}}\mathbf{U}$
People detection
Inverse height map transformation
CALIBRATION
$^{\mathcal{C}}\mathbf{x}_i^j$
Temporal pairing of observations
$\{\mathbf{x}_n \leftrightarrow \{\mathbf{x}'_m\}\}$
Initial pairwise mappings search
$\hat{\mathbf{M}}_{\mathrm{R}}$
Initial global extrinsics and targets
$\mathbf{M}_{\mathrm{R}}^j,\ ^{\mathcal{G}}\hat{\mathbf{x}}_i$
Global optimization
$\mathbf{M}_{\mathrm{R}}^j,\ ^{\mathcal{G}}\hat{\mathbf{x}}_i$
Mapping estimation
$^{\mathcal{C}}\mathbf{x}_i^j$
$\mathbf{M}^j$ or TPS^j
SERVER
Global frame transformation
$^{\mathcal{G}}\mathbf{x}_i^j$
Data fusion and tracking

Fig. 1 Overview of our people tracking system, relevant modules and coordinate transformations. Multiple nodes generate measurements in a 2D plan-view coordinate system that are sent to the centralized server for data fusion and tracking. The system calibration module takes a set of observations from different nodes as input, and outputs the transformations that align the observations to the common coordinate frame. Dashed arrows refer to inputs from another node. Symbols are as in the text

Accurate mapping of sensor observations to global coordinate system is crucial for successful data association, data fusion as well as for track management. Inaccurate extrinsics and intrinsics, and depth measurement distortions cause systematic errors in the observations, and they must be taken into account when transforming them to the global frame. We use the factory calibration for the intrinsics and do not model the depth distortions explicitly. Instead, we use mappings that have more degrees of freedom than the rigid transformation to translate and rotate the points, and to compensate the linear and nonlinear measurement errors as well. Thus, we avoid calibrating the cameras beforehand but still obtain accurate alignment of the observations from different nodes.

The local extrinsics are defined in the nodes, either automatically or using external information about the camera's orientation and height. The (distorted) observations are transformed to the global frame on the server side, which holds the mappings (global extrinsics) for each node. The mapping calibration routine is performed offline using recorded observations. The method first uses a RANSAC-based method for finding the connected sensor pairs and their relative 2D rigid transformations in the plan-view domain. We construct a graph from the pairwise mappings where the nodes represent the global extrinsics of the sensors and the edges represent their connectivity. We transform the observations to the global coordinate frame, and similarly to 3D bundle adjustment methods [11]; we refine the mapping parameters and the positions of the observations using a global optimization routine. Finally, we re-estimate the local extrinsics with mappings that have more degrees of freedom compared to the rigid transformations and use the optimized observation positions as calibration target.

2.2 Plan-view transformation of depth maps

We make the conversion from the depth images to 3D point clouds using the standard pinhole camera model. Given a (non-homogeneous) 3D point $\mathbf{X} = (X, Y, Z)^\top$ in the camera coordinate frame, its projection to the image plane is defined by the camera intrinsics and the perspective projection: $\mathbf{u} = (f_u \frac{X}{Z} + c_u,\ f_v \frac{Y}{Z} + c_v)^\top$, where f_u and f_v are the camera focal lengths in horizontal and vertical dimensions, and $(c_u, c_v)^\top$ is the principal point. Accordingly, given a depth image pixel $\mathbf{u} = (u, v)^\top$, the corresponding 3D point at depth d is $(d(u - c_u)/f_u, d(v - c_v)/f_v, d)^\top$. A 3 × 3 matrix containing the camera intrinsics is denoted with **K**. We ignore the lens and depth distortion from the camera model and use the default factory calibration for the rest of the intrinsics.

The plan-view coordinate transformation is done by rotating the original point cloud with the camera orientation matrix **R** so that the floor plane becomes parallel to the depth image plane, and its distance from the camera origin equals the camera height. The resulting point cloud is rendered to a height map image using an orthographic projection, which transforms the points to height map image pixels coordinates. The point depths are subtracted from the camera height h, such that the floor level equals zero and the targets' heights appear as positive pixel values in the resulting height map image. A height map pixel is represented as a homogeneous 2.5D point ${}^{\mathcal{H}}\mathbf{U}$. The transformation of a homogeneous 2.5D point ${}^{\mathcal{D}}\mathbf{U} = (du, dv, d, 1)^{\top}$ that corresponds to a depth image pixel at position $\mathbf{u} = (u, v)^{\top}$ and to a depth value d can be expressed as

$$
\begin{aligned}
{}^{\mathcal{H}}\mathbf{U} &= \mathbf{PRK}^{-1}\,{}^{\mathcal{D}}\mathbf{U} \\
&= \begin{bmatrix} \frac{s}{\Delta y} & 0 & 0 & -\frac{sx_0}{\Delta y} \\ 0 & \frac{s}{\Delta y} & 0 & -\frac{sy_0}{\Delta y} \\ 0 & 0 & -1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} r_1 & r_2 & r_3 & 0 \\ r_4 & r_5 & r_6 & 0 \\ r_8 & r_7 & r_9 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&\quad \times \begin{bmatrix} \frac{1}{f_u} & 0 & -\frac{c_u}{f_u} & 0 \\ 0 & \frac{1}{f_v} & -\frac{c_v}{f_v} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} du \\ dv \\ d \\ 1 \end{bmatrix},
\end{aligned} \tag{1}
$$

where **P** combines the orthogonal projection and the transformation to the height map image domain, s is the side length of the height map image (in pixels), x_0 and y_0 are the minimum x and y values of the top-view point cloud, and Δx and Δy are the ranges of the point cloud in x and y dimensions. In the above-mentioned, the height map image is assumed to be square, and the x and y dimensions are scaled with similar scale factor.

2.3 Mapping observations to global frame

With ideal depth camera intrinsics and local extrinsics, the transformation of the depth image pixels to the plan-view domain is distortion free, and the observations from different sensors are accurately aligned with rigid 2D transformations. Denoting a sensor measurement in the plan-view domain with a homogeneous 2D point ${}^{\mathcal{C}}\mathbf{x} = ({}^{\mathcal{C}}x, {}^{\mathcal{C}}y, 1)^{\top}$, the transformation to the global frame is defined by a 3×3 matrix $\mathbf{M}_{\mathrm{R}}$:

$$
{}^{\mathcal{G}}\mathbf{x} = \mathbf{M}_{\mathrm{R}}\,{}^{\mathcal{C}}\mathbf{x}, \tag{2}
$$

where prescripts $\mathcal{G}$ and $\mathcal{C}$ refer to the global and the local (plan-view) coordinate systems, respectively.

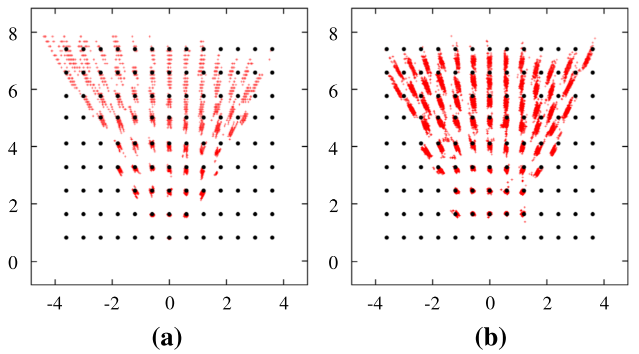

Fig. 2 Illustration of the measurement errors of a depth sensor from the top-view perspective. Black circles refer to paper cones with a radius of 10 cm and a height of 50 cm from the floor. Red dots refer to measurements of the cones in two experiments: **a** the sensor set to a height of 25 cm and directed towards the cones, **b** the sensor set to a height of 2 m with an approximate pitch angle of 30 degrees downwards. The cameras are located at the origin. The units of the plots are in metres

However, inaccurate intrinsics and local extrinsics, lens distortions and depth measurement errors introduce both linear and nonlinear errors, and a rigid mapping does not transform the points correctly. Figure 2 illustrates the problem. It shows two top-view images rendered from the depth maps that were captured from multiple paper cones placed on the floor. In the first image, the camera was parallel to the floor and directed perpendicularly to the cones. In the second image, the camera was raised and angled downwards. Both images show errors in scale, shear, projective errors as well as complex nonlinear errors.

Several depth camera models and calibration methods have been proposed that take into account the depth distortions (e.g. [3,13]). However, as our goal is to avoid explicit camera modelling and a manual pre-calibration step, we compensate for the errors directly in the plan-view domain by including an error model to the global extrinsics. For linear errors, we use rigid, similarity, affine and homography transformations, and replace $\mathbf{M}_{\mathrm{R}}$ in Eq. 2 with a corresponding 3×3 transformation matrix. Although the previous mappings can be solved with linear methods [11], we estimate them using a nonlinear optimization routine and minimize the point transfer error:

$$
\underset{\mathbf{M}}{\arg\min} \left\{ \sum_{i=1}^{n} \rho d \left({}^{\mathcal{G}}\hat{\mathbf{x}}_i, \mathbf{M}\,{}^{\mathcal{C}}\mathbf{x}_i \right)^2 \right\}, \tag{3}
$$

where **M** is any of the previous mappings, ${}^{\mathcal{G}}\hat{\mathbf{x}}_i$ are the known reference points, ${}^{\mathcal{C}}\mathbf{x}_i$ are the corresponding sensor measurements and $d(\cdot,\cdot)$ is the Euclidean distance between the points. As an initial guess for **M**, we use the rigid transformation $\mathbf{M}_{\mathrm{R}}$. The optimization problem is made more robust by using by a loss function ρ, for which we use Huber loss. To compensate linear and nonlinear errors, we use thin-plate spline

(TPS) mappings [6], which we briefly review here. The TPS mappings have been widely used in image warping, and also to model the distortions of depth cameras [3]. They produce smooth spatial mappings by minimizing the approximate curvature

$$I = \int\int \left[\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 f}{\partial y^2} \right)^2 \right] \mathrm{d}x\mathrm{d}y. \tag{4}$$

The TPS is an $\mathbb{R}^2 \to \mathbb{R}$ mapping that consists of an affine part and a nonlinear part defined by the TPS kernel ϕ. For a set of p control points $(c_{xi}, c_{yi}), i = 1, \ldots, p$ and associated weights $(w_1, w_2, \ldots, w_p)$, the TPS maps a point (x, y) as

$$f(x, y) = a_x x + a_y y + a_0 + \sum_{i=1}^{p} w_i \phi(r_i), \tag{5}$$

where $r_i = \|(c_{xi}, c_{yi}) - (x, y)\|$ and $\phi(r) = r^2 \log r$, and where the affine part is defined by a_x, a_y and a_0. To construct an $\mathbb{R}^2 \to \mathbb{R}^2$ TPS warp, two TPS mappings sharing the common control points $\mathbf{c}_i = (c_{xi}, c_{yi}, 1)^\top$ sampled from ${}^{\mathcal{G}}\hat{\mathbf{x}}_i$ are used, and the transformation of Eq. 2 can be written as

$${}^{\mathcal{G}}\mathbf{x} = \mathbf{M}_{\mathrm{A}}\, {}^{\mathcal{C}}\mathbf{x} + \mathbf{W}\Phi \left({}^{\mathcal{C}}\mathbf{x} \right), \tag{6}$$

where $\mathbf{W}$ is a $3 \times p$ matrix of column vectors $\mathbf{w}_i = (w_{xi}, w_{yi}, 0)^\top$ (w_{xi} and w_{yi} being the weights in x and y dimensions), and

$$\Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \ldots, \phi_p(\mathbf{x}))^\top$$

with

$$\phi_i(\mathbf{x}) = d(\mathbf{x}, \mathbf{c}_i)^2 \log(d(\mathbf{x}, \mathbf{c}_i)).$$

Thus, the complete TPS has $2 \times (3 + p)$ parameters.

For Eq. 4 to have a square integrable second derivatives, the TPS has the following constraints

$$\sum_{i=1}^{p} w_i = 0, \quad \sum_{i=1}^{p} w_i c_{xi} = 0 \quad \text{and} \quad \sum_{i=1}^{p} w_i c_{yi} = 0, \tag{7}$$

and a linear system for solving the coefficients of the TPS can be constructed:

$$\begin{bmatrix} \mathbf{K} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{0} \end{bmatrix}, \tag{8}$$

where $\mathbf{K}_{ij} = \phi(\|(c_{xi}, c_{yi}) - (c_{xj}, c_{yj})\|)$, the ith row of $\mathbf{P}$ is $(c_{xi}, c_{yi}, 1)$, $\mathbf{w} = (w_1, w_2, \ldots, w_p)^\top$ and $\mathbf{a} = (a_x, a_y, a_0)^\top$. A regularization parameter λ is added to the system to smooth the mapping in noisy circumstances. With $\lambda = 0$, the interpolation is exact, and the larger the value, the smoother the mapping. In practice, λ is incorporated into the system by replacing the matrix $\mathbf{K}$ with $\mathbf{K} + \lambda\mathbf{I}$, where $\mathbf{I}$ is a $p \times p$ identity matrix. We used a normalized value for λ and a scale factor of 1 as described in [10]. To reduce the computational burden, we avoid using all of the target points as control points for the TPS. Instead, we sample the control points from ${}^{\mathcal{G}}\hat{\mathbf{x}}_i$ using a 50 mm Euclidean distance threshold.

2.4 Initial calibration of the sensor network

The sensor network topology and the initial pairwise mappings between the sensors are solved using a RANSAC-based method. The first step is to find a set of point correspondences $S = \{\mathbf{x}_i \leftrightarrow \{\mathbf{x}'_j\}\}$ between the measurements of the first sensor and the second sensor. For each measurement $\mathbf{x}_i$, we search for a set of putative point correspondences L using the temporal proximity of the measurements as matching criteria. A point $\mathbf{x}'_j$ is added to L if the absolute difference between the timestamps of $\mathbf{x}_i$ and $\mathbf{x}'_j$ is less than a predefined limit (0.1 s in our experiments). From L, we search for clusters of spatially close observations, and from each cluster we save only the element with the smallest absolute time difference. With this approach, we avoid associating $\mathbf{x}_i$ to multiple measurements of the same person within the current time interval. On the other hand, the sensors do not necessarily observe the same persons, and thus, there may be multiple actual correspondences for each $\mathbf{x}_i$. We use a distance of 500 mm to cluster the observations belonging to L.

The initial pairwise mappings are estimated using RANSAC. For each iteration, we sample three point pairs from S, and estimate the rigid 2D transformation $\hat{\mathbf{M}}_{\mathrm{R}}$ between the point sets. Using $\hat{\mathbf{M}}_{\mathrm{R}}$, we transform the points $\{\mathbf{x}'_j\}$ to the first sensor's coordinate frame, and count the number of inliers, i.e. the points with a transfer error less than a predefined limit (300 mm in experiment 1 and 500 mm in experiments 2 and 3, see Sect. 3.2). If the inlier count is larger than a predefined threshold, we re-estimate $\hat{\mathbf{M}}_{\mathrm{R}}$ using all the inliers and re-calculate the number of inliers as to evaluate the fitness of the model. A larger number of inliers do not necessarily correlate with a better model, since it does not take into account the possibility of invalid matches that may occur if the sensors have seen different targets simultaneously. Therefore, each time the second inlier count increases, we calculate an additional score and accept the improvement in the model only if the additional score has increased as well.

The additional score is based on the assumption that the people which move in the overlapping area of a sensor pair are visible to both sensors. For the score, we calculate the convex hull of all measurements for each sensor, and find

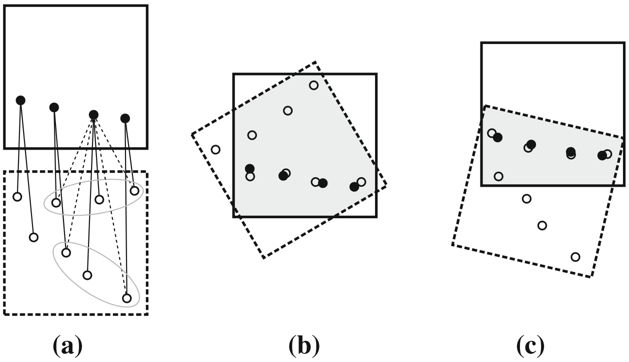

Fig. 3 Finding the initial pairwise mappings with a RANSAC-based procedure: **a** an initial set of point correspondences between the measurements of the first sensor (solid dots) and the measurements of the second sensor (open dots) are searched based on their temporal proximity (dashed lines). The candidates are spatially clustered (ellipses), and from each cluster, the measurement with the smallest time difference is saved (solid lines). **b** At each RANSAC iteration, the measurements and the view (dashed rectangle) of the second sensor are transformed to the first sensor's coordinate system using the current transformation hypothesis. The intersection of the sensor views is calculated (grey area), and the fitness of the model is evaluated by calculating the ratio of the inliers to the number of all point pairs that belong to the intersection. **c** The result from another iteration round. In contrast to **b**, there is only one outlier that belongs to the intersection, and the transformation hypothesis is given a higher score

the intersection. Then, the score is computed by calculating the ratio of inliers that belong to the intersection versus the number of all point pairs that belong to the intersection. The score could be evaluated in each RANSAC iteration, but we found it too slow for practical applications. Thus, we calculate the number of inliers first, and whenever it has increased, we calculate the additional score. In our experiments, it was approximately 100 times slower to evaluate the additional score compared to the plain inlier test. Figure 3 illustrates the approach. Given a confidence level, the number of iterations needed to obtain the correct model can be estimated using the standard method. For example, assuming that 5% of the point correspondences are inliers, approximately $n = \frac{\log(1-0.99)}{\log(1-0.05^3)} \approx 37{,}000$ iterations are required to find the correct mapping with a confidence level of 99%.

The number of inliers and the additional score can be used to determine which sensor pairs observe a common area, and thus are connected. In our experiments, we construct a two-element vector for each sensor pair consisting of the normalized number of inliers and the additional score. We use the k-means algorithm to classify the sensor pairs into groups of connected and disconnected pairs. To normalize the number of inliers, we divide the value with the area of the intersection.

From the connectivity information and the estimated pairwise mappings, we construct a graph where the vertices represent the sensors (the rigid transformations from the sensors' coordinate system to the global coordinate system), and the edges represent their connectivity. As we construct the graph, we select one of the sensors as the base sensor, and find the shortest paths from that one to each of the other sensors. For each sensor, we find the transformation that maps the sensor measurements to the base sensor's coordinate system by traversing the graph and combining the transformations at each node. The transformation of the base sensor is set to identity.

Our initial calibration method has some similarities with [24], which focused on estimating pairwise homographies and connectivity information in 2D camera networks. To constrain the homography estimation, the method used a guided sampling of temporally matched trajectories. The candidate homographies were given a score based on the number of inliers, and the homography with the highest score was used to validate the connectivity. A score similar to ours and a fixed threshold were used in the validation step. In contrast to the method, we do not make any assumptions about the observations in sampling and treat them equally. We include the validation step to the RANSAC loop which increases the likelihood for obtaining the correct mapping. Finally, we employ both the number of inliers and the additional score for determining the connectivity which yields more robust results.

2.5 Global optimization of the parameters

Lastly, we refine the initial global extrinsics with a global optimization routine. Similar to 3D bundle adjustment approaches [11], we not only optimize the mapping parameters (cameras in bundle adjustment) but the structure (i.e. the target positions in the global frame) as well. Thus, the goal is to find the estimates for the target positions and for the global extrinsics that minimize the global transfer error. Denoting the (homogeneous) 2D target points in the global frame with ${}^{\mathcal{G}}\hat{\mathbf{x}}_i$, and the corresponding observations made by sensor j with ${}^{\mathcal{C}}\mathbf{x}_i^j$, the solution is obtained by solving the following minimization problem

$$\underset{\mathbf{M}_{\mathrm{R}},{}^{\mathcal{G}}\hat{\mathbf{x}}}{\arg\min} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} v(i,j)\rho d\left({}^{\mathcal{G}}\hat{\mathbf{x}}_i, \mathbf{M}_{\mathrm{R}}^j\, {}^{\mathcal{C}}\mathbf{x}_i^j\right)^2 \right\} \tag{9}$$

where $\mathbf{M}_{\mathrm{R}}^j$ is the rigid transformation of the jth sensor and $v(i,j)$ is an indicator function that has value 1 if the point i has been measured by the jth sensor, and 0 otherwise. The measurements ${}^{\mathcal{C}}\mathbf{x}_i^j$ are the inliers resulting from the RANSAC method. Additionally, we complement the original inlier set with the points that have a larger transfer error (500 mm in experiment 1 and 2000 mm in experiments 2 and 3, see Sect. 3.2), and thus obtain more points for the optimization. We initialize the points ${}^{\mathcal{G}}\hat{\mathbf{x}}_i$ by transforming the measurements ${}^{C}\mathbf{x}_i^j$ to the base sensor's coordinate system using the initial global extrinsics and calculating the average position of the transformed points. The influence of outliers is reduced by using a loss function ρ, for which we use Huber loss. We solve the system using the Levenberg–Marquardt algorithm.

3 Experiments

3.1 System implementation

3.1.1 System set-up

In our people tracking system, the depth processing nodes are built upon Utilite PCs with ARM Cortex-A9 processors running at 1 GHz with 2 GB of RAM. We use Asus Xtion structured light depth cameras to obtain depth frames from the scene. The software is implemented in C++, and we use integer arithmetic and pre-calculated lookup tables whenever possible to speed up the processing time. The nodes are synchronized with the network time protocol (NTP), and the data is sent via the user datagram protocol (UDP) to the server. The data processing in nodes is single threaded so that one processing step from capturing a depth frame to sending the observations to the server takes less than 40 ms, and a frame rate of 25–30 FPS is achieved.

For the experiments, we recorded depth video sequences at a frame rate of 30 Hz and with a resolution of 320 × 240 pixels. We processed the depth frames offline using a standard laptop PC and the same depth processing pipeline implementation we used in the nodes. The calibration method was implemented in C++ with OpenCV [7] and Boost [22] libraries, and the nonlinear optimization problems were modelled and solved with the Ceres solver [1].

3.1.2 Detecting people from height maps

To detect people from depth frames, we use background subtraction which we apply in the height map domain. Depth cameras do not produce a uniform response from different types of surfaces and materials, and objects may appear noisy, contain holes and parts of them may flicker between consecutive frames. Thus, the distribution of height map pixel values in the background is bimodal, consisting of zero values (no depth observed) and values from the actual background. Although memory efficient and fast to compute, simple recursive filters are not optimal with bimodal distributions.

In our model, we maintain two buffers for each height map pixel: a short-term buffer and a long-term buffer. Both buffers have a length of five samples. We store the latest five pixel values in the short-term buffer. Once a predefined time interval Δt has elapsed, we update the oldest value of the long-term buffer with the maximum value of the short-term buffer. The background model is also updated by taking the median value of the long-term buffer. Thus, the update rate of the background model is determined by Δt. In our experiments, we used a short 10 s value for Δt.

The foreground image is extracted from the background model in a pixel-wise manner. Given a height map pixel value h and a corresponding background pixel value b, the resulting foreground height map is determined as follows:

$$h = \begin{cases} h & \text{if} \quad h \geq b + t \\ 0 & \text{otherwise} \end{cases}, \tag{10}$$

where t is a threshold margin (300 mm in our experiments). To remove speckle noise from the foreground, we apply a 2D median filter with a kernel size of 5 × 5 pixels.

We detect people from the foreground height map image by searching for local maxima with a search radius of approximately 40 cm. In our implementation, we divide the height map image into square windows of equal size, find local maxima from each window and store them in a list of candidates. For non-maxima suppression, we first sort the candidate list in ascending order by height. Then, we iterate through the list and compare each candidate with the remaining candidates. If the Euclidean distance between the candidate pair is less than the search radius, we remove the current candidate. The remaining list contains the observations of people. Finally, the observations are converted from the height map domain back to the plan-view domain.

For the experiments, we prune spatially redundant observations by binning them into equally spaced 100 × 100 mm bins, and keeping only the first values of each bin. This introduces a quantization error; however, it is relatively small compared to the noise level and depth quantization errors of the cameras that are used in our implementation.

3.1.3 Determining the plan-view transformation parameters

When floor pixels are visible, the cameras' local extrinsics can be obtained from the depth images. We assume that the cameras are installed such that their roll angles are zero, and that they are rotated around their lateral (pitch) axes only. Thus, the raw disparity map pixels that correspond to the floor plane can be detected using the v-disparity image technique [17]. From the depth image pixels that correspond to the floor, we determine the normal vector of the floor plane in the camera's coordinate system. The normal vector is used to construct the local extrinsic's rotation matrix $\mathbf{R}$, as well as the camera height h from the floor. As there may be several planes parallel to the ground plane in the scene (e.g. tables), we select the plane that is lowest. To populate the entries of matrix $\mathbf{P}$ in Eq. 1, we first create a point cloud that we transform with $\mathbf{R}$, and then search for the values x_0, y_0, Δx and Δy. We use a size of 200 × 200 pixels for the height map image.

3.2 Data sets

To evaluate the proposed method, we collected multiple data sets from three different sites and four camera configurations: Metro station entrance, Cafeteria 1 and 2, and Room. For Metro station entrance and Cafeteria configurations, we recorded two data sets: one using a mobile robot as the only visible target, and one from daily use with multiple people moving in the environment. For room configuration, we recorded a mobile robot run, and a set of RGB images to be used for the checkerboard-based calibration. For the robot data sets, we also recorded the ground truth trajectories. The data were collected as raw depth frames at a frame rate of 30 Hz and with a frame resolution of 320 × 240 pixels. Although we processed the frames offline using a standard laptop PC, we used the exact same implementation that is used for real-time operation on the depth camera nodes. Figure 4 illustrates the camera configurations 1–3 and the corresponding ground truth trajectories. The details of each configuration are presented below.

3.2.1 Ground truth from robot measurements

To collect ground truth observations, we attached a Styrofoam ball of 40 cm diameter to a mobile robot. The ball was installed at a height of 150 cm using a white plastic pipe. To obtain ideal data, we manually cleaned the robot observations of false positives and turned off the spatial pruning of observations. The robot was remotely controlled, and it was driven at a constant speed of 0.5 m/s. To determine the ground truth position, the robot was equipped with wheel odometry and a Hokyo UTM-30LX scanning range finder providing data with a 40 Hz scan rate. First, an occupancy grid map was created with [16]. After that, the robot pose was estimated using the adaptive Monte Carlo localization (AMCL) [14] with initial position set manually. Both algorithms are available in [20].

3.2.2 Configuration 1: Metro station entrance

The first configuration consisted of six cameras that were attached to the ceiling of a metro station entrance and directed downwards. The installation height of the cameras was 6.5 m, and the floor area covered by the cameras was approximately 140 m^2. The cameras were organized in a grid of 3 rows and 2 columns, and the overlap of the camera frustums was from 0.5 to 1 m at the floor level. The spacing between the cameras varied from 4 to 6 m. The real-life data set was acquired over 30 min. After spatial pruning and temporal matching the observations between the camera pairs, the remaining

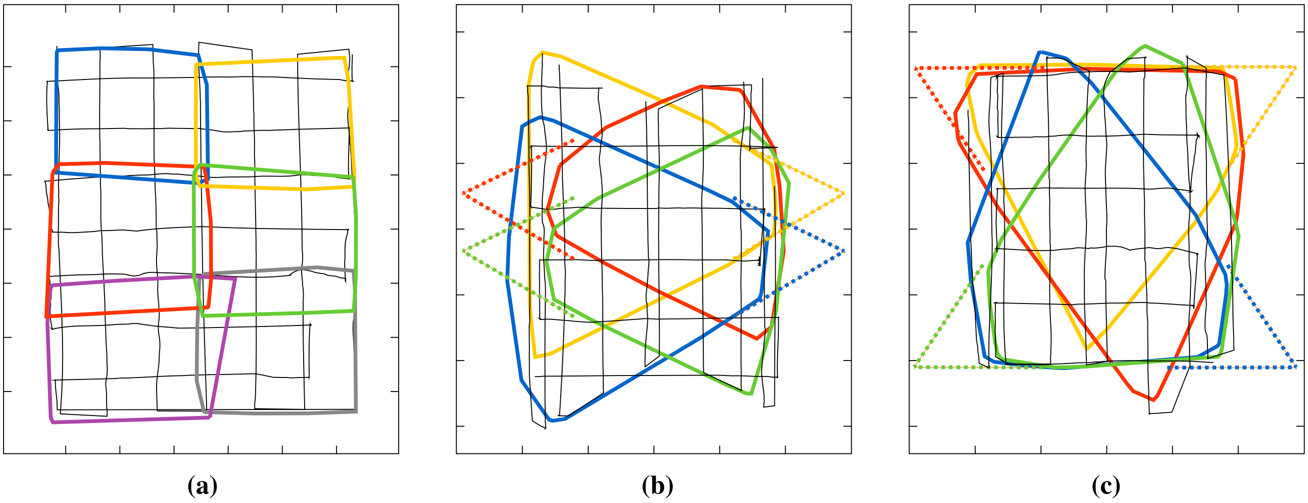

Fig. 4 The camera configurations 1–3 and their view frustums illustrated from the top-view perspective. **a** The configuration 1 included six cameras attached to the ceiling in a grid, and directed downwards. **b** The configuration 2 included four cameras aligned pairwise in parallel. The camera pairs were directed to the opposite sides of the space with a downward pitch angle of approximately 30 degrees. **c** The configuration 3 was similar to the configuration 2, but the cameras were placed in the corners of the area and directed to the centre. In each plot, the solid black line refers to the ground truth trajectory visible to any of the cameras. In **b**, **c**, the dashed triangles refer to the cameras' position and field of view. In **a**, there are a total of 15 possible sensor pair combinations in which 3 are horizontally connected (aligned in x-axis), 4 vertically connected, 4 diagonally connected and 4 disconnected. The tick marks are in 2 m intervals

number of observations varied from 184 to 709 (mean 464) per sensor, and the total number of individual observations (i.e. the points ${}^{\mathcal{G}}\hat{\mathbf{x}}_i$ in Eq. 9) was 1334. The robot data had 501–1028 observations per sensor (mean 706) and 1971 individual targets. The real-life data had 172 TPS control points, the robot data 146 and the ground truth data 530. Figure 4 also illustrates the vertically, horizontally and diagonally connected sensor pairs that we refer to in the later sections of this paper. The main direction of the traffic was along the vertically connected sensors (along the y-axis in Fig. 4).

3.2.3 Configuration 2: Cafeteria 1

The second camera configuration consisted of four cameras which were attached to the walls of a cafeteria of approximately 10 × 12 m. The cameras were aligned pairwise on opposite sides of the area such that the distance between the camera pair was 1.5 m, and their optical axes were parallel. The cameras were installed at a height of 2.3 m, and aimed downwards at approximately 30 degrees. The site had multiple tables limiting the free movement of people, and the tables were moved to collect the robot data. The real-life data set was acquired over 60 min. After spatial pruning and temporal matching the observations between the camera pairs, the remaining number of observations varied from 1329 to 1578 (mean 1406) per sensor, and the total number of individual observations was 2315. The robot data had 3006–3874 observations per sensor (mean 3438), and 5718 individual targets. The real-life data had 1242 TPS control points, the robot data 1057 and the ground truth data 1356.

3.2.4 Configuration 3: Cafeteria 2

The third configuration was similar to the second, but we moved the cameras to the corners of the cafeteria and directed them towards the centre. The real-life data set was acquired over 60 min. After spatial pruning and temporal matching the observations between the camera pairs, the remaining number of observations varied from 1578 to 1831 (mean 1689) per sensor, and the total number of individual observations was 2761. The robot data had from 3922 to 5325 observations per sensor (mean 4259), and 6975 individual targets. The real-life data had 1247 TPS control points, the robot data 1155 and the ground truth data 1262.

3.2.5 Configuration 4: Room

In the fourth configuration, three cameras were installed to the ceiling of an empty room of size 7.5 × 4.0 m. The installation height of the cameras was 2.6 m, and the angle approximately 25 degrees downwards. The cameras were set to three corners of the room, and aimed so that the longer sides of the room were completely covered by the cameras. For the data set, the robot was driven along a 4 × 7 grid path spanning the room completely. After the pruning, the data from different sensors contained from 832 to 1057 observations. In contrast to the other data sets, the outliers (mainly false positives due to noise) were not removed from the data. In the experiments, we used a threshold of 2000 mm for the outliers.

3.3 Evaluation procedures

3.3.1 Initial calibration

In the first set of experiments, the goal was to evaluate the robustness of the RANSAC-based method to find the initial pairwise mappings, and thus the ability of the method to find the network topology. We used configuration 1 for the experiments, as it had both connected and disconnected sensor pairs. We randomly selected 5, 10 and 30 min sequences from the data set and ran the method for 5000, 20,000 and 50,000 iterations. Each experiment was repeated 20 times. For each experiment, we stored the global extrinsic matrix, the number of inliers, the area of the overlap between the sensor views and the quality score defined in Sect. 2.4. For each pairwise mapping, we defined a two-element feature vector consisting of the number of inliers relative to the overlap area, and the mapping quality score. Then, we applied the k-means algorithm on the feature vectors to classify the mappings in two groups: connected and disconnected sensor pairs. Finally, we compared the classification results to the ground truth.

There are two types of errors that can occur during the classification procedure: a disconnected sensor pair may be classified as connected (we call this error type A) and a connected pair may be classified as disconnected (error type B). Error type A is critical since it can lead to a wrong network topology, and the people tracking becomes unpredictable. Error type B is less severe. In the case of such error, an edge is absent from the graph, and depending on the network topology, it may be compensated by traversing the graph along another path. A typical example of such a scenario is when two sensors have a large overlap and a third sensor has a small overlap with the first sensor and a large overlap with the second sensor. The first and third sensors may be classified as disconnected, but a path can still be constructed via the second sensor. From the classification results, we calculated the number of times error types A and B occurred.

To evaluate the precision of the initial pairwise mappings, we calculated the standard deviations of rotation angles between the mappings and the absolute differences between the smallest and largest angles. Additionally, we created histograms of the angle deviations. We analysed the horizontally, vertically and diagonally connected sensor pairs separately.

3.3.2 Experiments with different mappings

The goal of the second set of experiments was to compare and analyse how different mappings perform with the proposed auto-calibration method, and to compare our approach to typical checkerboard-based calibration method. In experiments 1–4, we first evaluated how the mappings compensate for systematic errors in the ideal case and calibrated the system using the ground truth as reference data. Then, we applied the auto-calibration method and calibrated the system using first robot data and second real-life data, and compared the results to both the ground truth and the estimated robot trajectory. In experiment 5, we calibrated the camera system using a planar checkerboard-based method and used the results in our system for the evaluation. The details of the experiments are presented below.

Experiment 1 The sensors were calibrated using the ground truth data points as calibration targets and sensor observations as input. Similarity, affine and homography transformations were obtained by solving Eq. 3, using the rigid transformations $\mathbf{M}_{\mathrm{R}}$ as initial estimates. The TPS mapping was solved using Eq. 8. The goal of the experiment was to analyse how different mappings compensate for systematic errors. The sensor observations were transformed using the estimated mappings, and the accuracy and precision of the transformed points relative to the ground truth were computed.

Experiment 2 The sensors were calibrated using the auto-calibration method and the robot data as input. The results were evaluated by transforming the sensor observations to the common coordinate frame using the estimated mappings, and comparing the transformed observations to the ground truth. A similarity transformation between the common coordinate frame of the sensors and the coordinate frame of the ground truth was applied before comparison. The goal of the experiment was to analyse the absolute performance of the auto-calibration method compared to the ground truth.

Experiment 3 The sensors were calibrated as in experiment 2. The results were evaluated by transforming the sensor observations to the common coordinate frame using the estimated mappings, and comparing the transformed observations to the estimated robot trajectory (${}^{\mathcal{G}}\hat{\mathbf{x}}_i$ in Eq. 9). To make the results comparable with experiments 1 and 2, the errors were scaled with the scale factor of the similarity transformation solved in experiment 2. The goal of the experiment was to evaluate how precisely the observations from different nodes can be aligned in a common coordinate system, which is crucial in data association and tracking tasks.

Experiment 4 The sensors were calibrated using the auto-calibration method and the real data as input. The results were evaluated as in experiment 3. The goal of the experiment was to evaluate how precisely the observations from different nodes can be aligned in a common coordinate system using real-life data as input.

Experiment 5 The goal of the experiment was to compare our method to conventional planar checkerboard-based calibration approach. One of the sensors was selected as a base sensor, and the transformations between the base sensor and the other sensors were estimated using OpenCV's stereo calibration routine [7]. The coordinate transformation between the floor plane and the base sensor was estimated by placing the checkerboard calibration target to several positions on the floor, and computing the average camera height and pitch and roll angles from corresponding images. The calibration was conducted from the RGB images of the sensors using the resolution of 1280 × 960 pixels. The intrinsics and lens distortions of the RGB cameras were calibrated beforehand. The factory settings for both depth sensor intrinsics and the coordinate transformations between the depth cameras and the RGB cameras were used. For the calibration, a 4 × 5 checkerboard target with the square size of 14 cm was used. The resulting camera poses were decomposed to local and global extrinsics to be used in our system. For the evaluation, the observations were first transformed to common coordinate frame by using the checkerboard-based calibration, and then, the results were transformed to robot coordinate frame by using the rigid and the similarity mappings.

Table 1 The number of horizontally and vertically connected camera pairs classified as disconnected (error type B), and their relative ratio to ground truth in parentheses ($n = 140$ for each experiment)

Iterations (k)	Sequence length		
	5 min	10 min	30 min
5	11 (7.9%)	19 (13.6%)	9 (6.4%)
20	6 (4.3%)	9 (6.4%)	6 (4.3%)
50	4 (2.9%)	8 (5.7%)	7 (5.0%)

For each experiment, we calculated the mean absolute errors and the standard deviations relative to the reference data (ground truth in experiments 1, 2 and 5, and reconstructed robot trajectory in experiments 3 and 4). As the errors are not Gaussian, we also estimated their distributions using the kernel density estimation (KDE) method, and calculated the 75% and 90% confidence regions of the error distributions (experiments 1–4). Finally, we calculated the ratio of observations with transfer error less than 40 cm of the reference data (experiments 1–4).

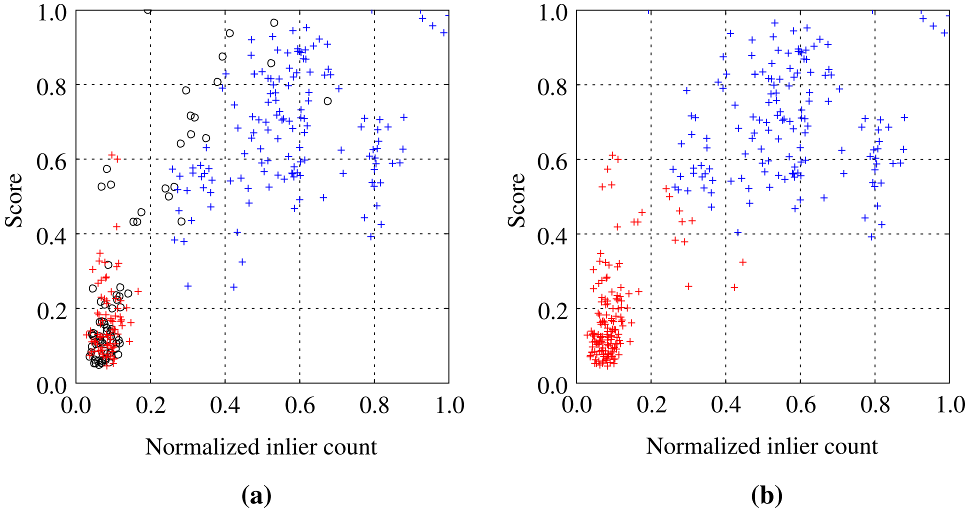

Fig. 5 An example of clustering the initial pairwise mappings. (a) The ground truth clusters. Red crosses are the disconnected sensors, blue crosses are the horizontally or vertically connected sensors, and black circles are the diagonally connected sensors. The x-axis is the normalized inlier count, and the y-axis is the score defined in Sect. 2.4. (b) The result of clustering with the k-means algorithm. Red crosses are the sensor pairs classified as disconnected, and blue crosses are the sensor pairs classified as connected. The coordinate axes are the same as in plot (a)

Table 2 Robustness and accuracy of the initial calibration

	Connection type		
	Horizontal	Vertical	Diagonal
n	489 (540)	692 (720)	144 (720)
SD	1.2	2.7	13.5
Δ	6.9	25.8	52.1

The first row shows the number of successfully estimated pairwise mappings and the ground truth in parenthesis. The second row shows the standard deviations of the rotation angles of the estimated transformations. The third row shows the range of the rotation angles. The angles are in degrees

4 Results

4.1 Initial calibration

Table 1 shows the number of horizontally and vertically connected sensor pairs that were erroneously classified as disconnected (error type B). Due to very small overlap between the sensor views, the diagonally connected pairs were typically classified as disconnected. To emphasize the results for adequate sensor overlaps, we omit the diagonally connected pairs from the table. The ratio of erroneously classified results to the ground truth varies from 2.9 to 13.6% in different experiments. Typically, best results were obtained with 50,000 iterations, but the results were very similar to those with 20,000 iterations. The sequence length did not notably affect the results.

Whenever the sensor pairs were correctly classified as connected, the estimated mappings were typically correct as well. The success ratio was 100% for the horizontally and vertically connected pairs, and it failed 2 times out of 145 with the diagonally connected pairs. The method is also robust to error type A. In the experiments, the disconnected sensor pairs were only classified as connected 4 times out of 360 (2 disconnected pairs, 9 experiments and 20 runs each). All errors occurred with the 5 min sequence: once with 5000 iterations, once with 20,000 iterations, and twice with 50,000 iterations.

Figure 5 shows an example of the classification results for an experiment with a 30 min data set and 50,000 iterations. The disconnected sensor pairs form a relatively tight cluster in the scatter plot, whereas the connected pairs are more widely spread. However, the two clusters are easy to separate with the k-means clustering algorithm. As stated above, the diagonally connected pairs are typically classified as non-overlapping pairs (error type B).

To evaluate the precision of the initial calibration routine, we calculated the standard deviations of the rotation angles of the pairwise rigid transformations. Table 2 shows that for the horizontally connected pairs, the standard deviation was smallest at 1.2 degrees. For the vertically connected pairs, the standard deviation was 2.7 degrees and for the diagonally connected pairs, it was 13.4 degrees. Table 2 also shows the number of correctly classified sensor pairs and the maximum absolute differences between the rotation angles. Figure 6 shows the distributions of the angle deviations as histograms. The horizontally connected sensor pairs had most of their values within ± 2 degrees, and the distribution is quite symmetric. The vertically connected pairs had most of their values within 2–4 degrees, and the distribution is skewed to the left. The diagonally connected pairs had no clear peak, and the distribution had no clear shape.

4.2 Experiments with different mappings

The results of experiments 1–5 are shown in Table 3. The table contains the mean absolute errors and the standard deviations of the sensor measurements transformed to the global frame using the estimated mappings and compared to the ref-

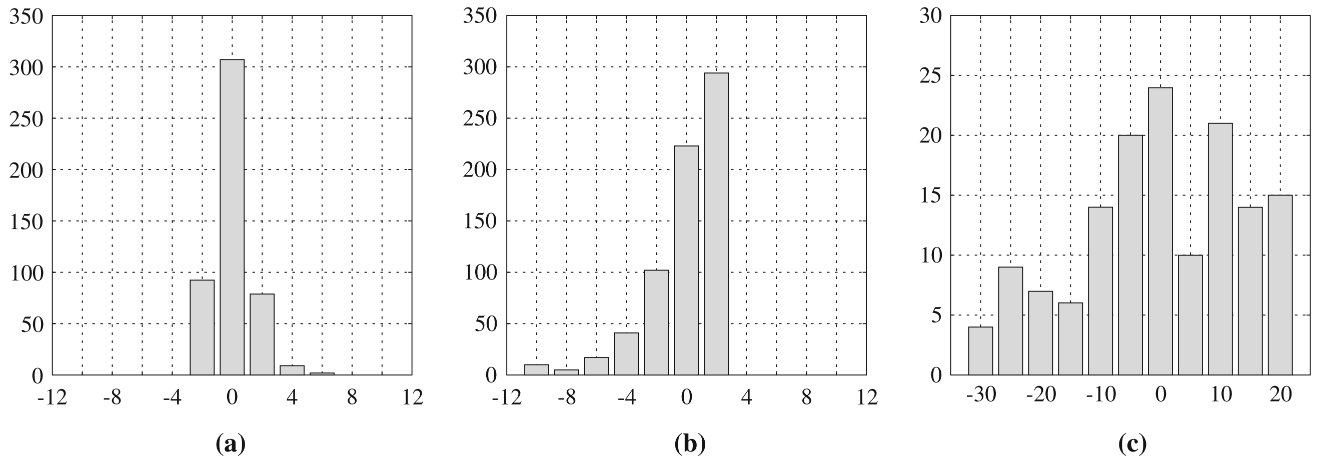

Fig. 6 Deviations of the rotation angles of the initial pairwise rigid transformations. **a** Horizontally connected sensor pairs, **b** vertically connected sensor pairs, **c** diagonally connected sensor pairs. In **a**, **b** the histogram bin size is 2 degrees and in **c** the histogram bin size is 5 degrees. In **b**, the smallest values are found in the − 10 degrees bin. See Table 2 for the actual range

Table 3 Mean absolute errors and standard deviations (in parentheses) for the experiments evaluating different mappings

	TPS	Homography	Affine	Similarity	Rigid
Configuration 1					
Experiment 1	0.113 (0.064)	0.156 (0.080)	0.160 (0.080)	0.167 (0.080)	0.405 (0.245)
Experiment 2	0.225 (0.121)	0.221 (0.118)	0.225 (0.126)	0.246 (0.153)	0.275 (0.210)
Experiment 3	0.062 (0.046)	0.082 (0.055)	0.087 (0.059)	0.100 (0.069)	0.114 (0.087)
Experiment 4	0.197 (0.102)	0.207 (0.111)	0.211 (0.113)	0.212 (0.118)	0.216 (0.125)
Configuration 2					
Experiment 1	0.101 (0.066)	0.242 (0.150)	0.245 (0.147)	0.255 (0.143)	0.574 (0.356)
Experiment 2	0.248 (0.199)	0.299 (0.261)	0.303 (0.223)	0.293 (0.178)	0.344 (0.224)
Experiment 3	0.092 (0.066)	0.127 (0.080)	0.171 (0.093)	0.212 (0.109)	0.250 (0.147)
Experiment 4	0.137 (0.080)	0.147 (0.086)	0.187 (0.102)	0.221 (0.123)	0.257 (0.160)
Configuration 3					
Experiment 1	0.101 (0.065)	0.210 (0.155)	0.235 (0.176)	0.247 (0.191)	0.637 (0.453)
Experiment 2	0.186 (0.130)	0.235 (0.193)	0.245 (0.223)	0.249 (0.224)	0.339 (0.261)
Experiment 3	0.102 (0.076)	0.178 (0.115)	0.196 (0.130)	0.207 (0.130)	0.282 (0.170)
Experiment 4	0.168 (0.116)	0.199 (0.133)	0.211 (0.149)	0.216 (0.150)	0.291 (0.184)
Configuration 4					
Experiment 1	0.152 (0.186)	0.158 (0.157)	0.193 (0.178)	0.239 (0.241)	0.491 (0.303)
Experiment 2	0.162 (0.171)	0.191 (0.221)	0.191 (0.160)	0.222 (0.193)	0.234 (0.207)
Experiment 5	–	–	–	0.533 (0.461)	0.638 (0.436)

erence data (the ground truth in experiments 1, 2 and 5, and the estimated robot trajectories in experiments 3 and 4).

In experiment 1, using configuration 1, the TPS had a mean error of 0.113 m and a standard deviation of 0.064 m. The homography, affine and similarity transformations had slightly larger errors: 0.156, 0.160 and 0.167 m, respectively, and a standard deviation of 0.080 m. The rigid transformation had an average error of 0.405 m and a standard deviation of 0.245 m. Using configurations 2 and 3, the errors for the TPS were similar, but for the homography, affine and similarity transformations the errors increased from approximately 50–80 mm. For the rigid transformation, the errors increased more: 0.169 m using configuration 2, and 0.232 m using configuration 3. Using configuration 4, the errors for the TPS slightly increased, probably due to the outliers that were not removed from the data.

In experiment 2, the results for all the mappings were quite similar using configuration 1, but different using configurations 2 and 3. Using configuration 2, the errors were in the range of 0.248 m (the TPS) to 0.344 m (the rigid transforma-

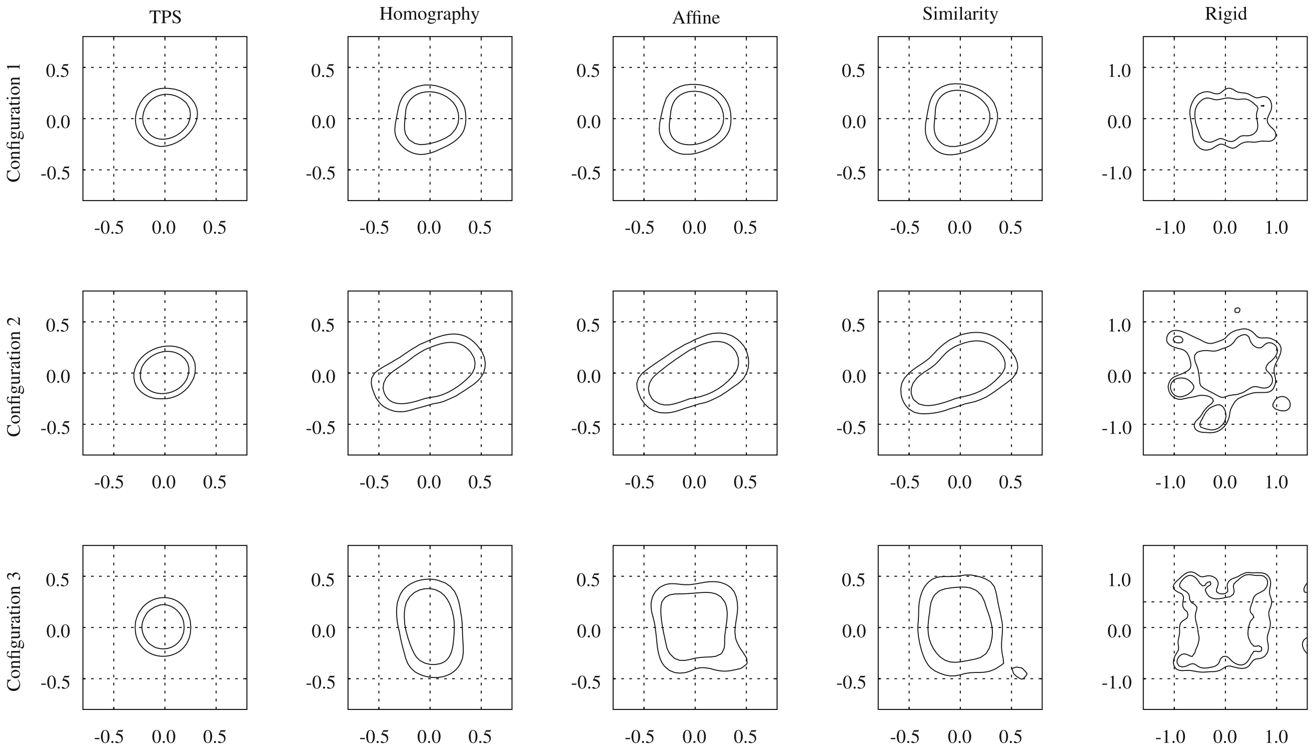

Fig. 7 The error distributions of the sensor observations transformed to the global frame and compared to the ground truth for experiment 1. The error distributions were estimated using the KDE method. The outer contours refer to the 90% confidence region and the inner to the 75% confidence region. The units are in metres

Table 4 The share of transformed observations with an absolute error less than 40 cm compared to the reference data

	TPS (%)	Homography (%)	Affine (%)	Similarity (%)	Rigid (%)
Configuration 1					
Experiment 1	99.9	99.8	99.7	99.6	54.4
Experiment 2	90.1	92.0	90.4	88.3	84.1
Experiment 3	99.9	99.8	99.8	99.6	99.2
Experiment 4	95.5	94.0	93.3	92.6	91.7
Configuration 2					
Experiment 1	99.9	85.8	85.3	84.3	40.0
Experiment 2	85.1	79.3	77.5	77.4	63.7
Experiment 3	99.9	99.4	97.5	95.0	86.7
Experiment 4	99.2	98.6	96.5	90.4	81.7
Configuration 3					
Experiment 1	99.5	91.8	86.8	86.1	38.7
Experiment 2	95.6	88.5	87.1	87.3	69.5
Experiment 3	99.5	95.8	93.2	93.2	81.3
Experiment 4	96.0	92.6	90.8	90.5	78.9

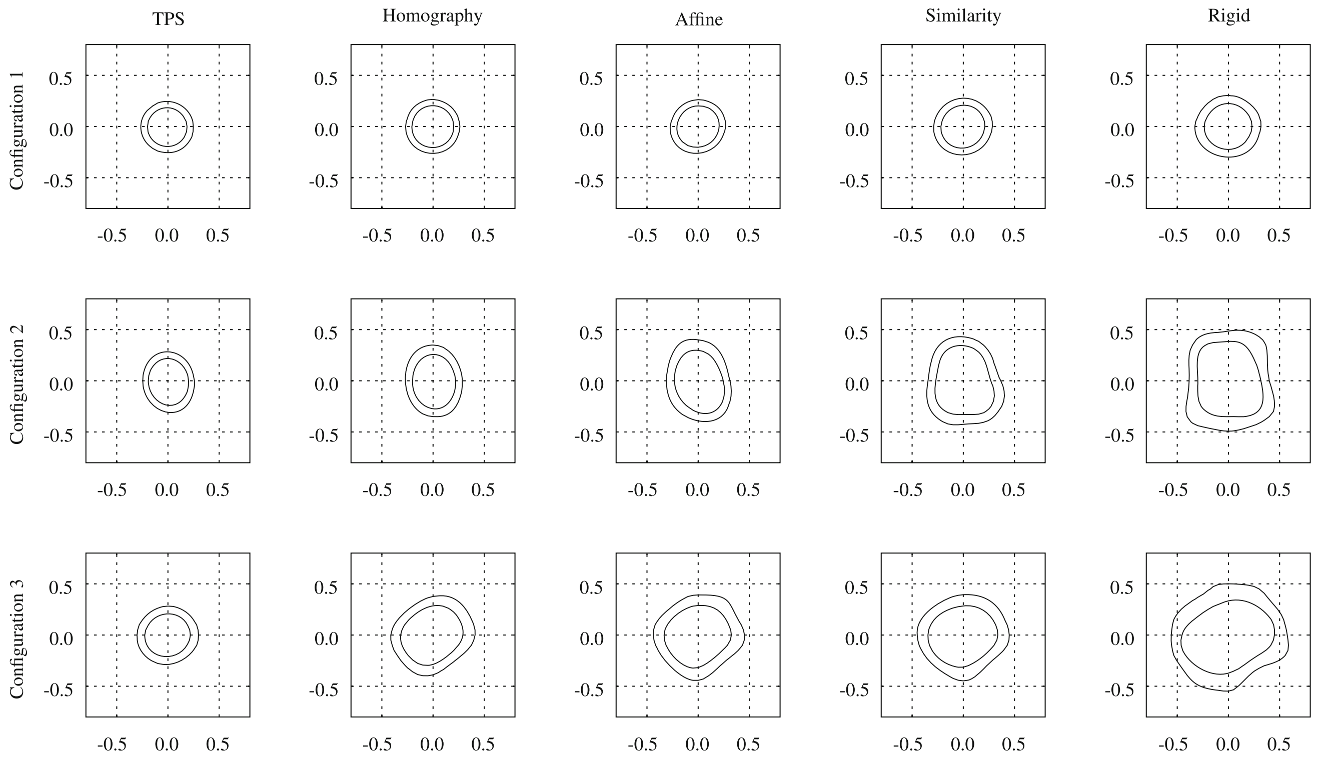

Fig. 8 The error distributions of the sensor observations transformed to the common coordinate frame and compared to the estimated robot trajectory for experiment 3. The error distributions were estimated using the KDE method. The outer contours refer to the 90% confidence region and the inner to the 75% confidence region. The units are in metres

tion), and the standard deviations were approximately 0.2 m. Using configuration 3, the errors were in the range of 0.186 (TPS) to 0.339 m (rigid transformation). The results for the configuration 4 were near similar.

In experiment 3, the TPS had the smallest errors: 0.062 m using configuration 1, 0.092 m using configuration 2 and 0.102 m using configuration 3. As in experiments 1 and 2, the rigid mapping had the largest errors, which varied from 0.114 m using configuration 1 to 0.282 m using configuration 3. The standard deviations varied similarly. For the TPS, the standard deviations ranged from 0.046 m using configuration 1 to 0.076 using configuration 3, and for the rigid transformation from 0.087 m using configuration 1 to 0.170 using configuration 3. As in experiments 1 and 2, the results for the other mappings were between the results of the TPS and the rigid transformation. The results for experiment 4 behaved similarly to experiment 3, but were slightly larger in general.

In experiment 5, the observations were first transformed to common coordinate frame using the results from the checkerboard-based calibration, and then to the robot coordinate system using rigid and similarity mappings. Table 3 shows that with the rigid mapping, the results are in line with experiment 1. Due to depth measurement errors, inaccurate (factory) intrinsics and depth-RGB calibration, the checkerboard-based calibration does dot properly align the observations from different sensors. The final mean absolute differences to ground truth were 0.638 m for the rigid mapping and 0.533 m for the similarity mapping.

Figures 7 and 8 visualize the error distributions of experiments 1 and 3 using the KDE method. The plots show that using the TPS, approximately 90% of the transformed measurements were within ± 0.25 m of the reference points (ground truth in experiment 1 and reconstructed trajectories in experiment 3). Using the other mappings, the errors were similar or slightly more spread using configuration 1 and larger using configuration 2 and 3. Also, using configurations 2 and 3, the shapes of the distributions were less symmetric in both the x and y dimensions for mappings other than TPS.

Table 4 shows the ratio of transformed observations with an absolute error less than 40 cm. Although the quantity does not separate the noise from the systematic errors, the table shows that errors decreased with mappings that have more degrees of freedom, and that TPS outperformed other mappings in 11 out of 12 cases. For experiment 1, the results of the TPS varied from 99.5 to 99.9%, the homography transformation from 85.8 to 99.8%, the affine transformation from 85.3 to 99.7%, the similarity transformation from 84.3 to 99.6% and the rigid transformation from 38.7 to 54.4%. The values for experiment 2 were lower, and, e.g. using the TPS, they varied from 85.1 to 95.6%. However, the exception is the rigid transformation as it had better results for experiment 2.

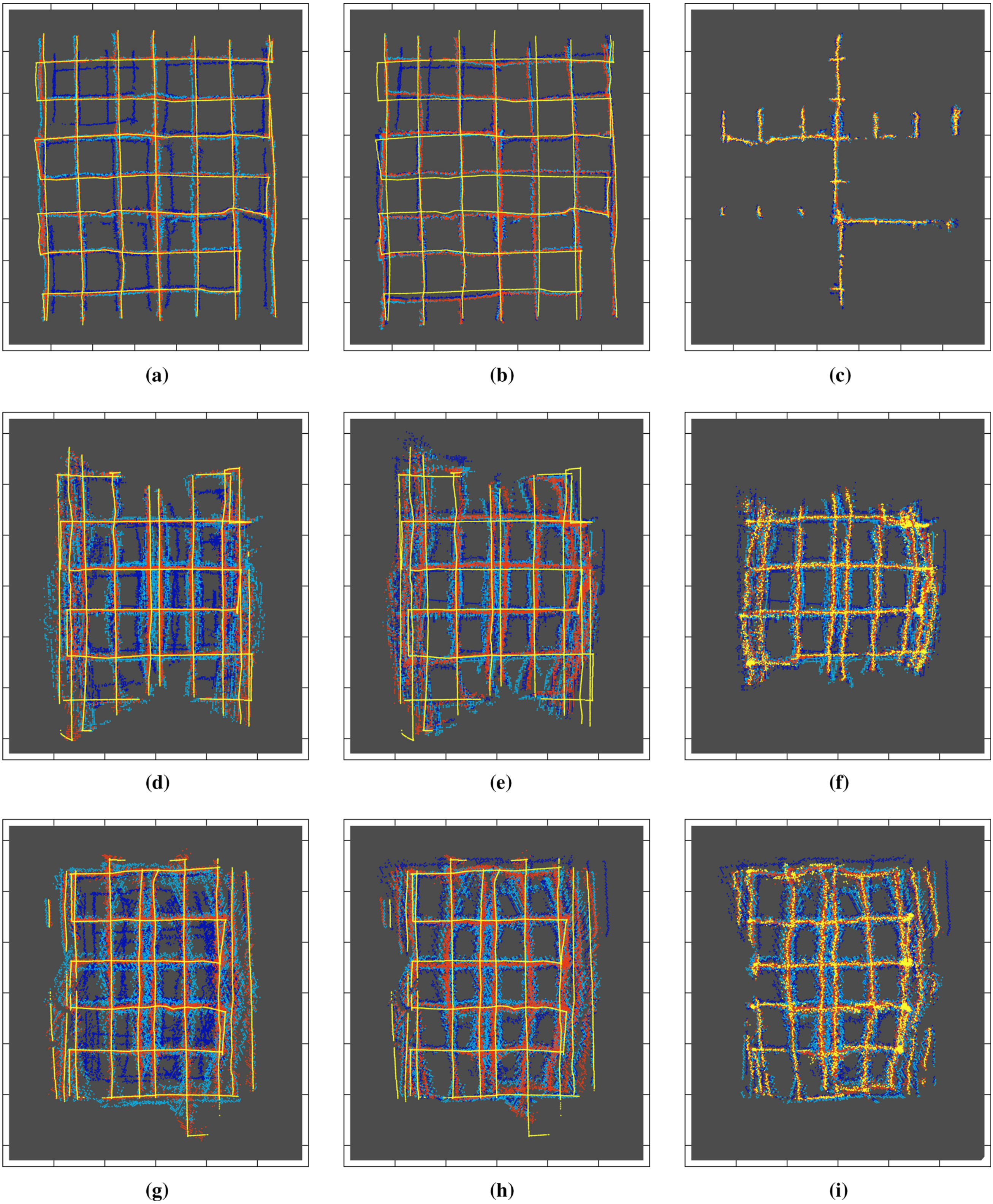

Fig. 9 Plan-view plots of the transformed points in experiment 1 (left column), in experiment 2 (middle column) and in experiment 3 (right column). The top row plots were generated using configuration 1, the middle row using configuration 2, and the bottom row using configuration 3. Yellow dots are reference points: ground truth in experiments 1 and 2, and the estimated robot trajectory in experiment 3. Dark blue dots are observations mapped using the rigid transformation, light blue using the affine transformation and orange using the TPS. The tick marks represent 2-metre intervals (color figure online)

In experiment 3, TPS had similar results compared to experiment 1. The results were better with the other mappings. In general, the results of experiment 4 were a few percentage points lower compared to the results of experiment 3.

Figure 9 illustrates the results from the top-view perspective. In the plots, the sensors' observations are mapped to the global coordinate system using the rigid, affine and TPS transformations. The reference points are included as well. Plots (a), (d) and (g) show the results of experiment 1. Due to measurement errors, the rigid transformations could not accurately align the observations with the ground truth. The affine transformation performed well using configuration 1, but using configurations 2 and 3, it suffered from nonlinear measurements errors. The TPS compensated for the errors in all cases, and the observations were accurately aligned with the ground truth. Plots (b), (e) and (h) show the results of experiment 2. All the mappings performed relatively well using configuration 1. The distortions of configurations 2 and 3 were worse, and none of the mappings could accurately align the observations with the ground truth. Plots (c), (f) and (i) show the results of experiment 3. The experiment shows that although the resulting mappings do not accurately align the measurements with the ground truth, the results are precise, and the observations from different nodes are aligned with each other.

5 Discussion and conclusion

We proposed an auto-calibration method for embedded depth camera networks using the plan-view approach for people tracking applications. The method enables fast installation of such systems as it automatically solves both the network topology and the transformations that map the sensor measurements to a common coordinate system. The approach is able to compensate for not only linear errors that originate from inaccurate intrinsics and local extrinsics, but nonlinear errors caused by lens distortions and depth measurement errors as well. The method is suitable for sensor networks which have both small and large pairwise camera overlaps.

We evaluated the method using Asus Xtion depth cameras. Since the method does not make any assumptions about how the depth data are generated, it can be used with other kinds of depth imaging devices. We used a simple and fast people detection algorithm based on background subtraction and local maxima search, but naturally the proposed approach can be used with any other people detection method as long as the observations are made in or converted to the plan-view domain.

To our knowledge, there are no prior studies that report solving the depth camera network topology and estimating the sensors' extrinsics simultaneously. Previous approaches apply affine transformations to align the observations from different sensors, which are not able to compensate for nonlinear errors. Our experiments show that homography and TPS mappings, especially, outperform the affine transformations in terms of accuracy and precision when aligning observations from multiple nodes. Other prior auto-calibration methods for depth camera networks aim to find full rigid 6-DoF transformations between the cameras, and assume that the cameras' intrinsics, lenses and depth distortions are modelled and calibrated beforehand.

Compared to conventional planar checkerboard-based calibration routine, our method has several advantages. The checkerboard calibration target has to be visible to two or more sensors simultaneously which limits how the cameras can be configured to the scene, especially when the angle between the optical axes of the cameras is large. Without careful pre-calibration of the depth sensors' intrinsics and depth distortions, the checkerboard-based methods are likely to suffer from misalignment errors. In our experiments, we used factory settings for the depth camera intrinsics and depth-RGB calibration, and the checkerboard-based calibration method was not able to deliver satisfactory results. In [19], a post-processing method for refining the checkerboard-based calibration was proposed. In the approach, it was assumed that one person only is moving in front of the cameras, and its average trajectory was used as a reference target in the refinement process. In contrast to our method, the procedure does not compensate for nonlinear depth measurement distortions.

Depending on the input data and the size of the camera overlap, RANSAC-based matching may lead to missed or false initial pairwise sensor mappings. We search for point correspondences between sensor pairs based only on their temporal proximity, and in some configurations the solution may not be unique. Thus, the robustness of finding initial pairwise mappings could be improved by using other metrics for the observations as well. Examples are geometric features such as size or shape of the target, dynamic features such as speed, or image-based properties such as texture or intensity histogram of the infra-red image typically available from depth sensors. Including additional features would enable using variants of RANSAC such as PROSAC (progressive sample consensus) [9] for faster and more reliable convergence.

Since we do not make any assumptions about the targets' movement or depth camera characteristics, the measurement errors are accumulated in the reconstructed trajectories. Thus, although the alignment of measurements from different sensors is precise, they may not accurately align with the ground truth. We suspect that results with less distortion could be obtained by assuming, e.g. straight trajectories for moving people and including the trajectory parameters and the TPS mapping parameters into the global optimization routine.

Acknowledgements Open access funding provided by Technical Research Centre of Finland (VTT). The authors would like to thank professor Tapio Takala from Aalto University, Finland, for valuable comments, and Alain Boyer from VTT Technical Research Centre of Finland for language revision.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Agarwal, S., Mierle, K., et al.: Ceres solver. http://ceres-solver.org
2. Basso, F., Levorato, R., Menegatti, E.: Online calibration for networks of cameras and depth sensors. In: The 12th Workshop on Non-classical Cameras, Camera Networks and Omnidirectional Vision, OMNIVIS (2014)
3. Belhedi, A., Bartoli, A., Gay-bellile, V., Bourgeois, S., Sayd, P., Hamrouni, K.: Depth correction for depth camera from planarity. In: Procedings of the British Machine Vision Conference, vol. 2012 (2012). https://doi.org/10.5244/c.26.43
4. Bevilacqua, A., Stefano, L., Azzari, P.: People tracking using a time-of-flight depth sensor. In: 2006 IEEE International Conference on Video and Signal Based Surveillance (2006). https://doi.org/10.1109/avss.2006.92
5. Beymer, D.: Person counting using stereo. In: Proceedings of the Workshop on Human Motion (HUMO'00), HUMO'00. IEEE Computer Society, Washington, DC (2000)
6. Bookstein, F.: Principal warps: thin-plate splines and the decomposition of deformations. IEEE Trans. Pattern Anal. Mach. Intell. **11**(6), 567–585 (1989). https://doi.org/10.1109/34.24792
7. Bradski, G.: The opencv library. Dr. Dobb's Journal of Software Tools (2000)
8. Camplani, M., Paiement, A., Mirmehdi, M., Damen, D., Hannuna, S., Burghardt, T., Tao, L.: Multiple human tracking in RGB-depth data: a survey. IET Computer Vision (2016). https://doi.org/10.1049/iet-cvi.2016.0178
9. Chum, O., Matas, J.: Matching with prosac—progressive sample consensus. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05) (2005). https://doi.org/10.1109/cvpr.2005.221
10. Elonen, J.: Thin plate spline editor—an example program in C++. https://elonen.iki.fi/code/tpsdemo
11. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (CUP), Cambridge (2004). https://doi.org/10.1017/cbo9780511811685
12. Harville, M.: Stereo person tracking with adaptive plan-view statistical templates. Image Vis. Comput. **22**, 127–142 (2002)
13. Herrera, C.D., Kannala, J., Heikkila, J.: Joint depth and color camera calibration with distortion correction. IEEE Trans. Pattern Anal. Mach. Intell. **34**(10), 2058–2064 (2012). https://doi.org/10.1109/tpami.2012.125
14. Howard, A., Gerkey, B.: Adaptive Monte-Carlo localization (AMCL) package, robot operating system (ROS). http://wiki.ros.org/amcl
15. Ikemura, S., Fujiyoshi, H.: Real-time human detection using relational depth similarity features. Lect. Notes Comput. Sci. (2011). https://doi.org/10.1007/978-3-642-19282-1_3
16. Kohlbrecher, S., von Stryk, O., Meyer, J., Klingauf, U.: A flexible and scalable SLAM system with full 3D motion estimation. In: 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (2011). https://doi.org/10.1109/ssrr.2011.6106777
17. Labayrade, R., Aubert, D., Tarel, J.P.: Real time obstacle detection in stereovision on non flat road geometry through v-disparity representation. In: Intelligent Vehicle Symposium, 2002. IEEE (2002). https://doi.org/10.1109/ivs.2002.1188024
18. Miller, S., Teichman, A., Thrun, S.: Unsupervised extrinsic calibration of depth sensors in dynamic scenes. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (2013). https://doi.org/10.1109/iros.2013.6696737
19. Munaro, M., Basso, F., Menegatti, E.: OpenPTrack: open source multi-camera calibration and people tracking for RGB-D camera networks. Robot. Auton. Syst. **75**, 525–538 (2016). https://doi.org/10.1016/j.robot.2015.10.004
20. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software, vol. 3 (2009)
21. Ruan, M., Huber, D.: Calibration of 3D sensors using a spherical target. In: 2014 2nd International Conference on 3D Vision (2014). https://doi.org/10.1109/3dv.2014.100
22. Schling, B.: The Boost C++ Libraries. XML Press, New York (2011)
23. Spinello, L., Arras, K.O.: People detection in RGB-D data. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (2011). https://doi.org/10.1109/iros.2011.6095074
24. Stauffer, C., Kinh, T.: Automated multi-camera planar tracking correspondence modeling. In: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings (2003). https://doi.org/10.1109/cvpr.2003.1211362
25. Svoboda, T., Martinec, D., Pajdla, T.: A convenient multi-camera self-calibration for virtual environments. PRESENCE Teleoper. Virtual Environ. **14**(4), 407–422 (2005)
26. Tian, Q., Zhou, B., Zhao, W.H., Wei, Y., Fei, W.W.: Human detection using hog features of head and shoulder based on depth map. J. Softw. (2013). https://doi.org/10.4304/jsw.8.9.2223-2230
27. Tseng, T.E., Liu, A.S., Hsiao, P.H., Huang, C.M., Fu, L.C.: Realtime people detection and tracking for indoor surveillance using multiple top-view depth cameras. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (2014). https://doi.org/10.1109/iros.2014.6943136
28. Wang, X.: Intelligent multi-camera video surveillance: a review. Pattern Recognit Lett. **34**(1), 3–19 (2013). https://doi.org/10.1016/j.patrec.2012.07.005
29. Wu, S., Yu, S., Chen, W.: An attempt to pedestrian detection in depth images. In: 2011 3rd Chinese Conference on Intelligent Visual Surveillance (2011). https://doi.org/10.1109/ivsurv.2011.6157034
30. Xia, L., Chen, C.C., Aggarwal, J.K.: Human detection using depth information by kinect. In: CVPR 2011 WORKSHOPS (2011). https://doi.org/10.1109/cvprw.2011.5981811
31. Zhang, X., Yan, J., Feng, S., Lei, Z., Yi, D., Li, S.Z.: Water filling: unsupervised people counting via vertical kinect sensor. In: 2012 IEEE 9th International Conference on Advanced Video and Signal-Based Surveillance (2012). https://doi.org/10.1109/avss.2012.82
32. Zhang, Z.: A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. **22**(11), 1330–1334 (2000). https://doi.org/10.1109/34.888718

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Mr. Otto Korkalo is a Senior Scientist at VTT Technical Research Centre of Finland Ltd. He received his M.Sc. (Tech.) degree from Helsinki University of Technology, Finland, in 2004. He has over 10 years of experience in computer vision and tracking for various applications including augmented and virtual reality. He is currently pursuing the PhD degree in Computer Science in Aalto University, Finland. His current research interests include vision-based people detection and tracking, pose estimation, and depth sensing technologies.

Mr. Tommi Tikkanen is a Robotics Engineer at GIM Robotics. He is currently leading the development of obstacle detection and tracking systems for autonomous vehicles. He received his M.Sc. (Tech.) in Automation Technology from Aalto University, Finland, in 2014. From 2012 to 2015, he worked as a researcher at VTT Technical Research Centre of Finland, where he focused on depth image-based pedestrian tracking.

Mr. Paul Kemppi is a Senior Scientist at VTT Technical Research Centre of Finland Ltd. He received his M.Sc. (Tech.) degree from Helsinki University of Technology, Finland, in 2005. He has over 15 years of experience in indoor positioning, wireless systems, and various environmental sensing technologies. His current research interests include robotics software, computer vision, and machine learning.

Mr. Petri Honkamaa is a Senior Scientist at VTT Technical Research Centre of Finland Ltd. He received his M.Sc. (Tech.) in Computer Science from Lappeenranta University of Technology, Finland, in 1997. After the graduation, he started his career at VTT focusing first on video compression technologies. Later, his research topics included various computer vision and optimization algorithms. Currently, his research focus is on 3D tracking algorithms and augmented reality applications.