RGB-D Camera Network Calibration for 3D Model Reconstruction

João Ramiro

***Abstract*—Accurate and real-time 3D reconstruction of objects is an aspiration that has seen many developments in the past years. Many approaches use a multi RGB-D camera setup and thus require a precise estimation of the relative poses between all cameras. This paper presents an approach to calibrate a RGB-D camera network system using a known calibration object based on fiducial markers. Results show that the method yields an initialization that is close to the real poses. Afterwards, a joint multi-cloud registration method performs fine adjustments on the obtained poses to attain a more accurate outcome. We use the system to create a 3D face model that is then used to track the mouth of a person in an automatic human feeding application. The designed calibration framework is lightweight, flexible and scalable since it can be used in networks with several cameras, achieving full system calibration in a few minutes.**

***Index Terms*—Camera networks, 3D reconstruction, RGB-D data, Multi Cloud Registration**

I. Introduction

ONE of the most common ways of capturing what surrounds us is through images, initially they were gray scale, now they are in color. With times this technology has evolved reaching the point where we are today, where we have RGB-D cameras capable of capture both depth an color images to generate a 3D depiction of the world.

Since the launch of Microsoft Kinect, this cameras have gained popularity and have been applied in the most varied areas such as in manufacturing, teleimmersion, surveillance and even healthcare [1], due to their ability to identify, locate and track objects in its view [2].

To obtain 3D data of large areas and get a more complete representation of objects, it is possible to combine multiple RGB-D sensors to record the 3D reconstructions of a scene. For the reconstructions to be precise, the cameras positions and orientations must be very accurately known so that when merging 3D data from multiple cameras the result is a correct depiction of the world.

In this work we propose a method to acquire camera positions and orientations relative to all cameras, i.e. calibrate a RGB-D camera network. One of the primary objectives of this paper is to use the 3D reconstructions to track objects in a multiple camera system. Particularly we apply our 3D reconstructions to Feedbot [3], a robot arm application that feeds meals to people with lack of upper-body mobility.

The proposed goal of calibrating a camera network contains two steps. In the first one, using a known object, we obtain an initialization of the pose of each camera, position and orientation, by detecting the known object based on virtual reality markers, on the RGB image of each camera and solving a Least Squares Problem. The second step consists in using the 3D reconstructions originated from each camera to further improve the estimated camera poses, using a global optimization methodology. In particular, we build our solution on the Global Procrustes Analysis ICP (GPA-ICP) method described in [4], that registers point clouds simultaneously onto the same referential. We also developed a physical system in order to test our approach. This system is based on Intel ZR300 depth cameras that use Robot Operating System (ROS)(Robot Operating System) to transmit data into a main server where most of the computations are done.

The whole process yields accurate results, is flexible and can be applied to both large and small 3D camera networks to produce 3D reconstructions of scenes and objects, that can then be used to generate 3D models and to track objects across time.

This paper has the following main contributions:

- **Multi RGB-D Camera Network Calibration Method:** We supply a method that uses marker based RGB feature matching in multiple cameras, followed by a joint multiple point cloud registration method, to obtain the global poses of the full camera system. Its ROS and python packages will be available in this repository[1].
- **Acquisition of More Complete Models:** We provide a method to stream in real-time a 3D reconstruction of a scenario, using RGB-D cameras spread around it. Alternatively, the cameras can also be placed in such a way that full 3D models of faces and objects can be obtained.

The organization of this paper is as follows. The state of the art is described in II. This includes a brief description on various approaches for the camera network calibration problem and also on the registration of multiple point clouds. In III the main problem is formulated mathematically, reaching two independently solvable problems. The method to calibrate the network is presented in IV. The experiment results and analysis is presented in V. A real world face tracking application that uses the outputs of our method is described in VI. The developed camera network system is explained, both hardware and software wise in VII, and the conclusions on the work are presented in VIII.

II. State Of the Art

A. Calibration of camera networks

Many camera sensors networks with different architectures have been developed over the years and, consequently, multiple calibration approaches have come with them.

[1]https://github.com/DonHaul/MscThesis

Some systems such as [5]–[7] offer very precise integrated solutions but at great cost. This is not the objective of this study, for which this section comprises a review of related work that present fewer, simpler and less expensive cameras networks.

Many of the previous works, use a calibration object, an object that is known before hand, to create relation between cameras an calibrate the network. A sphere is used as the calibration object [8], [9], since it can be easily detected in depth or RGB images by performing thresholding and image segmentation, its center point can be estimatedt. Each camera must obtain several images where the sphere is present, to create matches between different cameras in order to find their relative poses.

Virtual calibration objects, which consist in non physical features such as bright lights and spots created by a laser or a LED can are also used in [10], [11]. By having this well lit and easily identifiable point on the RGB image it is possible to create matches between different cameras. This points must be moved to several places to generate a virtual calibration object in 3D space that with the use of epipolar geometry, can be utilized to estimate the camera poses.

People can also be used as an calibration object. By detecting and tracking people, their walking trajectories can be inferred in different cameras. Afterwards, the transformations that best fits the trajectories together is found as show by the authors in [12], [13].

Another alternative is the use of fiducial markers, which are easily identifiable objects, usually planar, that serve as a point of reference. By placing this markers in front of the cameras it is possible to obtain an initial rough estimate of the cameras pose in a world reference frame as proposed in [14], [15]. This poses must be refined in order to accurately represent the scenes, using the 3D data from the used RGB-D cameras it is possible to use Iterative Closest Point (ICP) to obtain a better calibration.

Structure from Motion (SfM) is also another approach to the camera calibration problem. The authors of [16] use keypoint feature matching and SfM to calibrate 480 RGB cameras, followed by Bundle Adjustment (BA) routine.

Most of the papers cited above use a coarse to fine approach to calibrate the camera network. First they use the keypoints on the scene or in a calibration object that can be easily discernible to get a rough estimate of the camera poses. Because this estimate is usually based on pairwise transformations, i.e. the pose between pairs of cameras, a second step must be performed to globally estimate the camera poses. In systems with RGB cameras this final refinement is mostly done through BA or other formulated problems that minimize the reprojection error of the whole camera network. When the network uses RGB-D cameras, the point clouds (depth data), can be used to fine-tune the results using ICP or multi cloud joint registration algorithms. Some of these algorithms will be described next.

B. Point Cloud Registration Approaches

Using the 3D data, i.e. point clouds, originated from different cameras/points of view it is possible to estimate poses that best align the clouds and consequently obtain the poses of the cameras those clouds came from. One of the most known point set registration methods is ICP [17], an iterative method, that at each iteration finds correspondences between points in different clouds to approximate one point cloud to the other. Since ICP was first developed, a lot of progress has been made towards capturing tridimensional data and with that, a higher demand for accurate and flexible methods capable of multiple point cloud registration method. As we know, ICP can only be used to register pairs of point clouds, so if we wanted to align multiple point clouds, we would have to use it sequentially between pairs of clouds which would generate error with each point cloud that is registered. An alternative to this method are global registration methods that align multiple point clouds at once.

One of this methods is presented in [4], where the authors take advantage of Generalized Procrustes Analysis (GPA), which is a well-known technique that provides a least-squares solution to the multiple point set registration problem, and embeds it in a ICP Framework, thus creating the GPA-ICP algorithm that automatically aligns views with partial point cloud overlaps between them. This algorithm will be further described in IV, and it will be used in the as the camera pose refinement step in this paper.

The Joint Registration of Multiple Point Clouds (JR-MPC) algorithm proposed in [18], also aligns multiple clouds by converting each point cloud to a realization of a Gaussian Mixture Model (GMM), thus casting the problem into a clustering problem. The formulated optimization problem is solved using Expectation Conditional Maximization (ECM), which finds the parameters of the GMMs and consequently the poses that relate the point clouds.

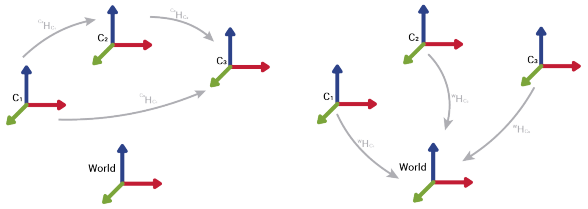
A hierarchical GMM implementation is presented in [19], where the point matching of different point clouds, is done over multiple scales as a recursive tree-based search thus increasing efficiency. The method is fast and yields an accurate point cloud registration.

The approaches that use GMM although accurate, only work when there is much overlap between clouds, which would cause point clouds with partial or no overlap to be mishandled. However, to some extent, this does not occur for GPA-ICP which because of that, it will be used as a refinement step in this paper.

III. Global Transformations from Pairwise Poses

A. Problem Description

The main problem we want to address here is the calibration of a camera network, i.e. we want a way of knowing the transformations of a camera, in relation to every other camera in the system. The aforementioned problem can also be generalized into what is shown in image 1 where, having a set of objects where we know the transformations between pairs of elements, we want to convert those object pairwise transformations into transformations relative to a single reference frame. The transformations we refer to are 3D rigid transformations, which contain a translation and a rotation component. They can be represented by



(a) (b)

Fig. 1: (a) Pairwise Transformations. (b) Global Transformations.

$$\overbrace{\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}}^{\boldsymbol{x_1}} = {}^{1}R_2 \overbrace{\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix}}^{\boldsymbol{x_2}} + {}^{1}t_2, \tag{1}$$

where ${}^{1}R_2$ is a 3×3 rotation matrix and ${}^{1}t_2$ a 3×1 translation vector that transform some point $\boldsymbol{x_2}$ in reference frame 2 into $\boldsymbol{x_1}$ in reference frame 1. If there are two sets of points X_1 and X_2 and the correspondences between points in each set are known, we are able to find the transformation in Eq.(1) by solving the Procrustes Problem [20] which boils down to solving

$$\begin{aligned} \hat{R}, \hat{t} = \quad & \underset{R,\,t}{\arg\min} \quad ||X_1 - \big(RX_2 + t\big)||_F^2 \\ & \text{subject to} \quad R \in SO(3), \end{aligned} \tag{2}$$

where SO(3) represents the set of 3D rotation matrices characterized by $R^TR = 1$ and det(R)=1.

However, in real world problems these point correspondences are usually not known. On top of that if we want to find the transformations between multiple point sets, the problem is converted in Eq.(3),

$$\begin{aligned} \underset{\alpha_{klij},\,{}^{w}R_i,\,{}^{w}t_i}{\arg\min} \quad & \sum_{i,j}^{N}\sum_{k,l}^{P} \alpha_{klij} || \, {}^{w}R_i X_{ik} + {}^{w}t_i - \big({}^{w}R_j X_{jl} + {}^{w}t_j\big)||_F^2 \\ \text{s.t.} \quad & {}^{w}R_i \in SO(3), \\ & i = 1,..,N \quad j = 1,..,N \quad k = 1,..,P \quad l = 1,...,P, \\ & \mathcal{F}(\alpha_{klij}) = 0, \end{aligned} \tag{3}$$

where N is the number of sets, P the number of points, ${}^{w}R_i$ and ${}^{w}t_i$ are the rotation and translation from each point set i into the world frame. X_{ik} is ith point of set k, and α_{klij} defines whether or not there is a correspondence between point k from set i and point i from set j. This correspondence variable may have some constraints, portrayed by $\mathcal{F}(\alpha_{klij}) = 0$ which represents some of the possible constraints to be applied such as, i.g. if a point can have one or multiples matches or if points are only a match when they are closer than a certain distance. As we can see, Eq.(3) is a non-convex problem due to its many constraints, making it a hard.

In this work we develop a method to tackle this hard problem. We use a known object exposed in Fig. 2, that contains a set of identifiable points, that can be easily found across all cameras. This means that variable α_{klij}, no longer needs to be found as it is previously determined by detecting equivalent points in different cameras. Moreover, by moving the calibration object in front of the cameras, it is possible to generate pairwise transformations, i.e. transformations between two cameras, which means that over time, rotations ${}^{j}R_i$ and translations ${}^{j}t_i$ between cameras i and j are acquired. This brings up the question: How can we this noisy transformations, to register all cameras on the same reference frame?

As we know, in many rigid transformations problems such as this one and the Procrustes Problem, the rotations can be found independently of the translations [20]. Because of this property this problem can be decoupled into a rotation and a translation problem described in Eq.(4) and Eq.(5).

$$\begin{aligned} {}^{1}\hat{R}_w, {}^{2}\hat{R}_w, ..., {}^{N}\hat{R}_w = \quad & \underset{{}^{i}R_w}{\arg\min} \quad \sum_{i,j}^{N} || \, {}^{i}\hat{R}_w - {}^{i}\hat{R}_j \, {}^{j}\hat{R}_w||_F^2 \\ & \text{subject to} \quad {}^{i}R_w \in SO(3), \\ & \qquad\qquad i = 1,..N \quad j = 1,..N \end{aligned} \tag{4}$$

$$\begin{aligned} {}^{1}\hat{t}_w, {}^{2}\hat{t}_w, ..., {}^{N}\hat{t}_w = \quad & \underset{{}^{i}t_w}{\arg\min} \quad \sum_{i,j}^{N} || \, {}^{i}t_w - \Big({}^{i}R_j \, {}^{j}t_w + {}^{i}t_j\Big)||_F^2 \\ & \text{subject to} \quad i = 1,..N \quad j = 1,..N \end{aligned} \tag{5}$$

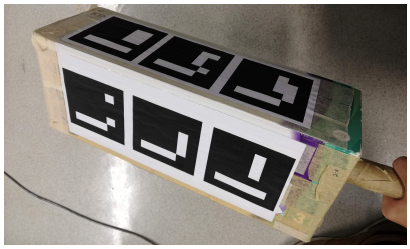


Fig. 2: Known Object.

The inputs in Eq.(4) and Eq.(5) are ${}^{i}R_j$ and ${}^{i}t_j$ which correspond to the pairwise rotation and translation between the objects with reference frames j and i. The minimization problem formulated for the rotations still has a constraint and thus it is a non-convex problem. The translations problem has a known format, the sum of the euclidean norms, which can be solved with the Least Squares (LS) method.

As stated before, we use a known object to obtain the camera transformations, so we first need to define that well-known calibration object so that we recognize correspondences between cameras.

This object needs to have easily identifiable features all around it so that cameras can identify it no matter what direction it is facing. The object in Fig. 2 is used in our method as it contains virtual reality markers, ArUco markers, that are easily detectable. To get its model, the transformations between markers must be known relative to each other. This will result in the exact same problem stated before: Having a set of pairwise transformations between objects, how can we have their transformation relative to a single reference frame? Being the same problem we can solve it with the same optimization problem, in order to obtain every marker in a world reference frame. Pairwise transformations are obtained as visualized in Fig. 3.

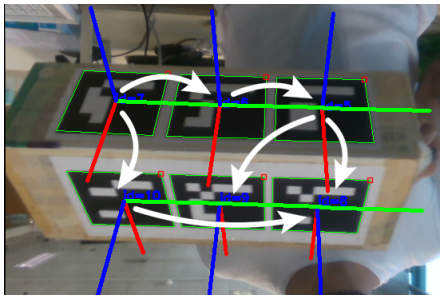


Fig. 3: Pairwise transformations within calibration object.

Having this pairwise transformations it is now imperative to solve the optimization problems in Eq.(4) and Eq.(5). The former is described in III-B while the latter is in III-C.

B. Solution for the Rotations

To address the rotations problem stated in Eq.(4), we first lift the constraint resulting in Eq.(6), which is a convex problem.

$$ {}^{1}\hat{R}_w, {}^{2}\hat{R}_w, ..., {}^{N}\hat{R}_w = \underset{{}^{i}R_w}{\arg\min} \sum_{i,j}^{N} || {}^{i}R_w - {}^{i}\hat{R}_j \, {}^{j}R_w ||_F^2 \quad (6) $$

Because of summation properties, this minimization can be formulated in a matrix form, where the term inside de frobenius norm can be written as Eq.(7)

$$ \underbrace{\begin{bmatrix} 0 & I & \dots & 0 \\ I & 0 & \dots & 0 \\ 0 & 0 & \dots & I \\ 0 & I & \dots & 0 \\ \vdots & & & \vdots \end{bmatrix}}_{I_0} \underbrace{\begin{bmatrix} {}^{1}R_w \\ {}^{2}R_w \\ \vdots \\ {}^{N}R_w \end{bmatrix}}_{Z} = \underbrace{\begin{bmatrix} {}^{2}R_1 & 0 & \dots & 0 \\ 0 & 0 & \dots & {}^{1}R_N \\ 0 & {}^{N}R_2 & \dots & 0 \\ 0 & {}^{3}R_2 & \dots & 0 \\ & \vdots & & \end{bmatrix}}_{E} \underbrace{\begin{bmatrix} {}^{1}R_w \\ {}^{3}R_w \\ \vdots \\ {}^{K}R_w \end{bmatrix}}_{Z} \quad (7) $$

This is the equality to solve where K is the amount of pairwise rotations, N the number of markers and we wish to obtain ${}^{i}R_w, \quad i = 1, 2, ..., N$. I_0 and E are $3K \times 3N$ matrices, where each 3 rows encode a pairwise rotation ${}^{j}R_i$. This is done adding a 3×3 identity matrix to the $3 \times j^{th}$ column of I_0 and add the rotation itself on the $3 \times i^{th}$ column of E, all the other elements of those 3 rows of I_0 and E will be 0. The matrices can then be rearranged into Eq.(8),

$$ (I_0 - E)Z = 0, \quad (8) $$

where it can be seen that the solution Z is the null space of $I_0 - E$. This equation system is overdetermined, i.e. $N > K$ on most use cases. On top of that, noise in the pairwise rotations will make it so that most the rows are linearly independent, since some of the obtained rotations will contradict each other e.g. ${}^{2}R_3 \neq {}^{3}R_4 \, {}^{4}R_2$. An overdetermined system where there are more linearly independent rows than there are columns results in an equation system with no solution Z that satisfies Eq.(8). In cases like this, the best solution will be one that approximates the null space of $I_0 - E$ shown in Eq.(9).

$$ \hat{Z} = \underset{Z}{\arg\min} \sum_{i,j}^{N} ||(I_0 - E)Z||_F^2 \quad \text{subject to} \ \ \text{rank}(Z) = 3. \quad (9) $$

This minimization problem can be solved by performing Singular value decomposition (SVD) on $I_0 - E$ resulting in $I_0 - E = U\Sigma V^T$. Then we extract the columns of V that have the least influence on generating $I_0 - E$, which will be the last three, resulting in $\hat{Z}$. Because we relaxed the problem by lifting its constraint in Eq.(6), the individual 3×3 matrices Z_i that make up matrix $\hat{Z}$ will not correspond to 3D rotation matrices.

Solving the orthogonal Procrustes problem, it is possible to project each 3×3 matrix onto the rotation matrix space, as shown in,

$$ \underset{{}^{w}R_i}{\arg\min} \quad || {}^{w}R_i Z_i^T - I||_F^2 \quad \text{subject to} \quad {}^{w}R_i{}^{T} \, {}^{w}R_i = I, \quad (10) $$

where $i = 1, 2, \dots N$. With this we finally get the rotations ${}^{w}R_i$ of all markers relative to world space.

C. Solution for the Translations

As stated before, Eq.(5) can be solved using the LS method in order to obtain the global translations between objects. This is done using the pairwise translations between them as input in conjunction with the rotations from III-B. Through summation properties and 3D transformation properties it is possible to convert the term inside the frobenius norm of Eq.(5) into:

$$ \underbrace{\begin{bmatrix} 0 & I & \dots & 0 \\ I & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ 0 & 0 & \dots & I \\ \vdots & & & \vdots \end{bmatrix}}_{I_l} \underbrace{\begin{bmatrix} {}^{w}t_1 \\ {}^{w}t_2 \\ \vdots \\ {}^{w}t_N \end{bmatrix}}_{v} = \underbrace{\begin{bmatrix} I & 0 & \dots & 0 \\ 0 & 0 & \dots & I \\ 0 & 0 & \dots & I \\ 0 & I & \dots & 0 \\ \vdots & & & \vdots \end{bmatrix}}_{I_r} \underbrace{\begin{bmatrix} {}^{w}t_1 \\ {}^{w}t_2 \\ \vdots \\ {}^{w}t_N \end{bmatrix}}_{v} + \underbrace{\begin{bmatrix} {}^{w}R_1 \, {}^{1}t_2 \\ {}^{w}R_N \, {}^{N}t_1 \\ {}^{w}R_N \, {}^{N}t_2 \\ \vdots \\ {}^{W}R_2 \, {}^{2}t_N \end{bmatrix}}_{b} \quad (11) $$

This is the equality to solve to obtain ${}^{i}t_w, \quad i = 1, 2, ..., N$. I_l and I_r are $3K \times 3N$ matrices and b is a $3K \times 3$ matrix. Each 3 rows of this matrices encode a pairwise translation ${}^{j}t_i$. This is done by adding a 3×3 identity matrix the $3 \times i^{th}$ column of I_l and in the $3 \times j^{th}$ column of I_l. The pairwise rotations itself is multiplied by ${}^{w}R_j$ and added in b all the other elements for those 3 rows of I_l and I_r will be 0. This system can be represented as,

$$ \underbrace{(I_l - I_r)}_{C} v = b. \quad (12) $$

Similarly to what occurs in III-B, noise will render the equation solutionless due to the same reasons as before. To obtain an approximate solution for this overdetermined system, the LS solution is calculated through the pseudo inverse:

$$ v = (C^T C)^{-1} C^T b. \quad (13) $$

v is then split up in 3×1 vectors which correspond to the best ${}^{w}t_i$ where $i = 1, 2, ..., N$ for this system.

In the end, having both the rotations from III-B and the obtained translations, we finally know the transformation of each object relative to a single reference frame.

D. Calibration Method

Using the techniques described in III-B and III-C, a method can now be implemented to obtain the model of the calibration object. In this method we use a camera to capture images of the model. At each captured frame the square markers are detected and using the positions of their corners, its is possible to estimate the transformation of each marker in the reference frame of the camera, by solving a Perspective-n-Point (PnP) problem as described in [21]. Combinations of this transformations are done to generate pairwise transformations between the markers. This process is repeated while we wish to obtain more pairwise rotations. Once there there are pairwise rotations connecting every maker to every other we can solve Eq.(7) to obtain the global rotations. If needed, after that more marker transformations are obtained from where pairwise translations are extracted. The translations are then computed and the marker poses relative to a single reference frame are obtain as in Fig. 1b.

It is important to note that due to the problem properties, it is possible to store information about the transformations in a compact manner, that remains the same size no matter how many transformations were calculated. If were to store $I_0 - E$ from the rotations formulation in Eq.(7), and $I_l - I_r$ and b from the translations in (11), those matrices would grow with each pairwise transformation detected. Since we want a lot of transformations, we can counteract this by storing at each time instant i for the rotations problem

$$B_i = (I_{0_i} - E_i)^T (I_{0_i} - E_i), \tag{14}$$

and for the translations problem

$$D_i = (I_{l_i} - I_{r_i})^T (I_{l_i} - I_{r_i}) \tag{15a}$$
$$y_i = (I_{l_i} - I_{r_i})^T b_i. \tag{15b}$$

It is then possible to sum B_i, D_i and y_i from all instants L and obtain the exact same results as in Eq.(7):

$$B = \sum_{i=1}^{L} B_i = (I_0 - E)^T (I_0 - E). \tag{16}$$

The same thinking is applied to D_i and y_i to obtain the matrices from Eq.(11):

$$D = \sum_{i=1}^{L} D_i = (I_l - I_r)^T (I_l - I_r) \tag{17a}$$
$$y = \sum_{i=1}^{L} y_i = (I_l - I_r)^t b. \tag{17b}$$

To obtain the rotations, instead of performing SVD$(I_0 - E)$, eigendecomposition is performed $B = Q\Lambda Q^{-}1$ where Q holds an eigenvector from which the last 3 columns, corresponding to the smallest eigenvalues, are extracted. Finally Z is split in 3×3 matrices and the orthogonal Procrustes problem is solved for each of them as displayed in Eq.(10). The translations are obtained by using formula in Eq.(18) which is equivalent to the pseudo inverse.

$$v = D^{-}1y \tag{18}$$

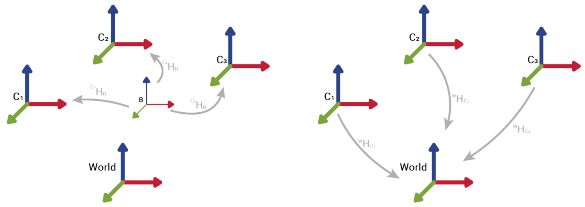
v is then split up in 3×1 to obtain the translations of each marker relative to the world just as described in III-C.

Having the marker poses in a world referential, it is possible to calculate where their corners are positioned in that same referential, which will be useful in posterior calculations. Since we know the marker side size, the corners are obtained by adding and subtracting $\frac{size}{2}$ in the x and y axis of each markers referential. With this we have obtained the full calibration object model which in the next section will be used to obtain the camera poses.

IV. Camera Network Calibration

A. Camera Pose Initialization

Although getting to the transformations of the cameras in the same referential may look difficult at first, in actuality the problem stated in III-A, can just as well be applied here. Having a set of cameras and their pairwise transformations it is possible to deduce their transformations into a world frame. This means that the equations and methods previously described, can also be used for the camera pose initialization. To get the transformations between cameras, they need to have in their view something in common, which in this case will be the calibration object from the previous section (Fig. 6). The current problem is described by image Fig. 4a, where B corresponds to the calibration object frame, and C_1, C_2 and C_3 will be each camera, that observe the object and know its transformation relative to their own reference frame.



(a) (b)

Fig. 4: (a) Cameras observing calibration object. (b) Global Transformations.

Pairwise transformations ${}^{C_j}R_{C_i}$ and ${}^{C_j}t_{C_i}$ are generated by composing transformations between cameras and the calibration object:

$${}^{C_j}R_{C_i} = {}^{C_i}R_B \, {}^{C_j}R_B{}^T \tag{19a}$$
$${}^{C_j}t_{C_i} = {}^{C_j}R_B{}^T(-{}^{C_j}R_B{}^T \, {}^{C_j}t_B) + {}^{C_j}t_B. \tag{19b}$$

The calibration process is analogous to the what is described in III-D, since it only differs on how the pairwise transformations are obtained. Previously, a single camera would visualize markers, where as now every camera in the system captures an image at synchronized time instants. Each camera locates, if present, markers in its image and uses them to estimate the transformation between the calibration object and itself by solving a PnP problem with the method described in [21]. The PnP problem uses the detected corner markers in each image, and its correspondence of the known 3D model to estimate a transformation of the object. This transformations

are then used to generate pairwise transformations between pairs of cameras. Afterwards, if enough transformations were generated, the rotations of each camera relative to the world is estimated. Subsequently, more images may be captured to get more pairwise transformations, followed by the estimation of the translations, thus obtaining the transformation of every camera relative to a single reference frame.

B. Camera Pose Fine Adjustments

After the initial camera pose calibration is performed, the results may not be accurate enough, due to factor such as, an inaccurate calibration object model, incorrect camera intrinsic parameters, inaccurate marker detection or unsynchronized cameras.

In order to fix the misalignment that might exist between cameras, another more fine calibration step is executed. This step consists in using the point clouds, i.e. the depth data originated from each camera, to perform the pose refinement. We use the GPA-ICP algorithm from [4], which combines GPA with ICP in order to achieve a global registration method for multiple point clouds. Because this method registers multiple point clouds simultaneously, it will distribute the errors evenly, as opposed to sequential registration approaches which registrate point clouds in pairs leading to the accumulation of error with each individual registration.

1) General Procrustes Analysis: GPA is a well-known technique that provides a least-squares solution when wish to find the transformations that best fit more than two points. Its equation, as shown in [22], is

$$\begin{aligned} \underset{R_i, c_i, t_i}{\arg\min} \quad & \operatorname{tr}\left(\sum_{i=1}^{m}\sum_{j=i+1}^{m}\left((c_i X_i R_i + 1_p t_i^T) - (c_j X_j R_j + 1_p t_j^T)\right)^T \right. \\ & \left.\left((c_i X_i R_i + 1_p t_i^T) - (c_j X_j R_j + 1_p t_j^T)\right)\right) \\ \text{subject to} \quad & R^T R = I, \end{aligned} \tag{20}$$

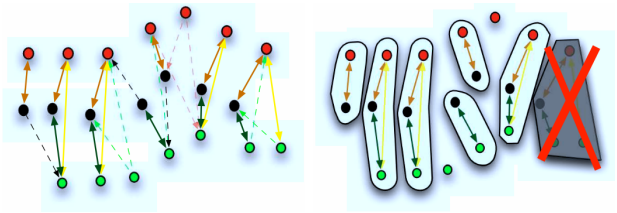
where $X_i, \quad i = 1, 2..m$ are sets of p points and k dimensions and c_i,R_i and t_i are the set's scale, rotation and translation relative to a world referential. 1_p is a vector of $p \times 1$ ones where p corresponds to the number of points in the sets. This optimization problem can be converted onto the formulation in Eq.(21) where $\hat{X}_i = c_i X_i R_i + j t_i^t$ as described in [4].

$$\sum_{i=1}^{m} \left\| \hat{X}_i - K \right\|^2 = \sum_{i=1}^{m} \operatorname{tr}\left(\left(\hat{X}_i - K\right)^{\mathrm{T}} \left(\hat{X}_i - K\right) \right) = \min \tag{21}$$

$$K = \frac{1}{m}\sum_{i=1}^{m} \hat{X}_i \tag{22}$$

In this new problem K is a $p \times k$ corresponds to the centroid of all point sets, and what we wish to find the R_i c_i and t_i that better fit each point cloud to that centroid. This equations assume that there exists matches between every point in every point set. One addition to make it also support missing data would be to have a $p \times p$ diagonal matrix M_i that defines for each set X_i if they have a corresponding point in the the centroid point set K. This is explained in detail in [4] and [22].

2) Algorithm: In order to register multiple point clouds, the authors in [4] combine GPA with a special point matching technique. They only consider as matches, groups that contain points from different point sets that are each other's closest neighbors, a relationship that the authors call of "mutually nearest neighbor". This is shown in Fig. 5a where the dotted line represents a one way neighbor relationship, while a full line represents a mutually nearest neighbor. Afterwards, mutual neighborships are chained together into groups, which are only considered valid if there is a maximum of one point per set in it as shown in Fig. 5b. From this it is possible to generate a matrix of correspondences that is updated at every iteration. This matrix is also important in the creation of a new centroid point set K in each iteration, by adding a point to it corresponding to the centroid of each individual group of mutual neighbors, causing points from those same groups to be driven towards the same position.



(a) (b)

Fig. 5: Point matching images taken from [4].(a)Nearest Neighbor vs Mutual Nearest Neighbor. (b)Independent Group Sets.

This matching method prevents points that are very far away from each other from compromising the results of GPA, leading to better point cloud matching. In conclusion algorithm results in the following steps:

1) Find the mutually nearest neighbors among point sets.
2) Define a centroid for each independent mutually nearest neighbor set.
3) Estimate transformations for each point cloud using the Extended Orthogonal Procrustes explained in [4].
4) Transform each point cloud using the estimated transformations.
5) Go back to step 1, until global convergence or the maximum number of iterations is reached.

Several experiments are performed, and the results are interpreted in V. In general the algorithm proposed in [4] yields accurate results as long as the initial point cloud poses have a proper initialization.

V. Experimental Evaluation

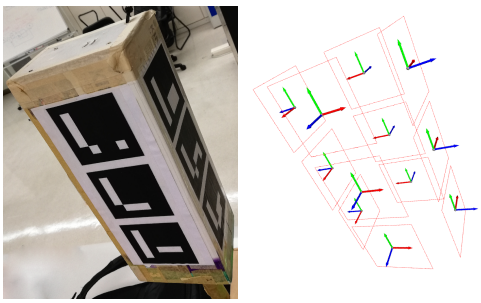
A. Calibration Object Experiments

We use a calibration object containing a set of ArUco markers, a well known fiducial marker that is detected and located using the OpenCV[2]. The markers are placed in every side of the calibration object shown in Fig. 2 so that multiple cameras, positioned at different standpoints, can find the calibration object using the markers detected in their view.

[2]http://bit.ly/opencvarucos

Furthermore each of the markers must have the same size length and be unique in order to distinguish them from each other.

To obtain the calibration object model, the object is placed and moved in front of a camera and the procedure stated in III-D should be followed. The resulting model is shown in Fig. 6b where the red line represents connects the corners of each marker, and each reference frame represents their position and orientation. We can see that the obtained model is similar to the actual object in Fig. 6a.



(a) (b)

Fig. 6: (a) Real Calibration Object. (b) Model of Calibration Object

This model can be compared to the real calibration object by measuring distances between corners of different markers. So to understand the correctness of the model, we measure the diagonal of each face of the calibration object, that is from the bottom left corner of the marker on the bottom, to the top right corner of the marker on the top, using a ruler with $\pm 0.0005\ mm$ uncertainty. Table I displays the measurements and how they compare. The values obtained from the model do not deviate more than $3\ mm$ from the ground truth.

TABLE I: Diagonal Measurements.

Side Diagonal [m]	1	2	3	4
Ground Truth	0.300	0.296	0.290	0.294
Model	0.303	0.298	0.292	0.296
Absolute Error	0.003	0.002	0.002	0.002
Relative Error	1.06%	0.64%	0.75%	0.67%

To be able to measure how well the method described in III-D performed, we use the self reprojection error Eq.(23) to calculate in an image, for all detected marker corners N, how close the detected corners with pixel coordinates p_i are from the ones reprojected from the model $\hat{p}_1$. Its equation is

$$\bar{E}_r = \frac{\sum_{i=1}^{N} ||p_i - \hat{p}_i||_F^2}{N}, \tag{23}$$

The corners this metric uses can be visualized in Fig. 9 where the corners detected in the image are in green and the ones reprojected from the model are in red.

Afterwards, to measure how well a camera can detect the calibration object using the built model, we measure the self reprojection error at each frame. In Fig. 7 we see how it evolves over time. From this graphic we conclude that the error is stable, which means that most of the time, the detected corners and the reprojected corners stay around the same pixel distance. The gap in the graphic refers to a period where no marker was detected, and consequently, no error was calculated.

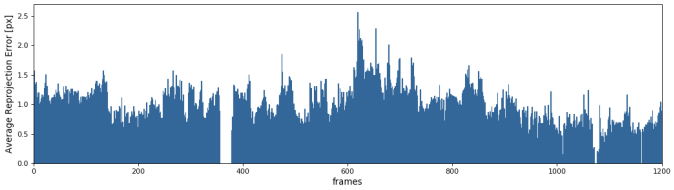


Fig. 7: Average Reprojection Error in one of the experiments.

By plotting the histogram for this data in Fig. 8, we see that the error distribution resembles a Gaussian distribution which has the mean $\mu = 0.98$px and a standard deviation of $\sigma = 0.31$px,which means that on average, each reprojected corner is 0.98 pixels apart from the corresponding detected corners, and from the standard deviation we can interpret that 99.7% (3σ) of the frames have a detection error between 0.05 and 1.91px.

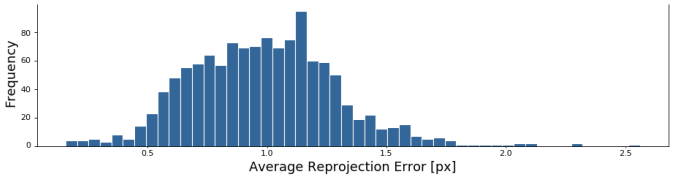
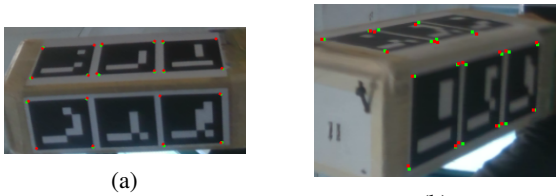


Fig. 8: Average Reprojection Error in one of the experiments.

Some outliers can also be distinguished on the histogram. These are the result of frames where corners of the detected markers are incorrectly estimated due to blur and/or to an unfavorable viewing angle. Specifically, frame 619 has the worst outlier. Fig. 9b shows a zoom of that frame where we see those two situations occurring. The bad marker detections leads to a less accurate estimation of the calibration object transformation, resulting in reprojected corners that are on average 2.56 pixels apart from the corresponding detections. When compared to the detection on frame 53, for example, with an error of 1.0 pixels, we can visualize how the effects described above influence the detection comparing to when they are absent in Fig. 9a.

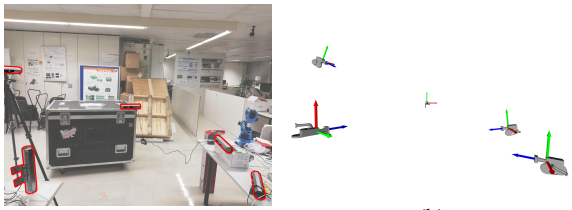


(a) (b)

Fig. 9: Detections and reprojections. (a) In frame 53 (b) In frame 619.

B. Experiments on Camera Poses

To obtain the camera poses we follow the procedure described in IV. After running the method we can see qualitatively in image 10 how the modeled camera poses on the right, match the real camera poses on the left.



(a) (b)

Fig. 10: (a) Real Camera Setup. (b) Obtained Camera Poses.

From this images we comprehend that the method produces poses similar to the real ones, but this results are greatly dependent on how accurately we know the calibration object and how it was moved in front of the cameras, as the faster is is moved and the further away it is, the worse will be the marker detections and consequently the final result.

C. Fine Tuning the Results

To improve the results obtained from the initialization step described above, GPA-ICP is utilized. The method takes as input a group of captured point clouds with the corresponding camera poses applied to them. In this case we use, the robot arm clouds in Fig. 12a. The resulting poses produce improved point cloud registratios as seen in the images on Fig. 12b.

To quantitatively see how much the results are improved with this step, we calculated we first measured the distances between every camera pair in a four camera setup using a metric tape with ± 0.0005mm uncertainty, this metrics were considered as the ground truth. We then generated 3 independent pose initializations, which will vary depending on factors such as the way we moved the calibration object or on the amount of pairwise transformations captured. We then also captured a point clouds from the scene in Fig. 11 where each camera will have in its respective point cloud a part of the robot arm.

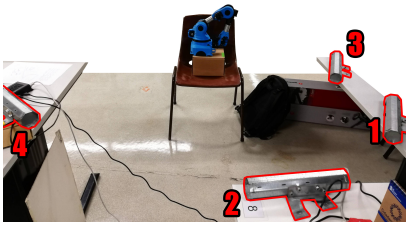


Fig. 11: Camera Poses Pointing at an robot arm.

The camera poses at the end of every step were then compared to the ground truth, leading to table II. C_i-C_j corresponds to metrics taken between camera i and camera j. This table displays the absolute and relative errors calculated for each initialization, before and after GPA-ICP was applied.

From it we can understand that with just the initialization, the obtained poses are erroneous and very dependent on how the calibration object moves since for one initialization a camera pair may have very little error, while in another the same pair will be very inaccurately estimated. In the average row we can see the average distance error between camera pairs, and from it we can comprehend that the average error for each initializations is very similar varying between 4.7cm and 5.8cm.

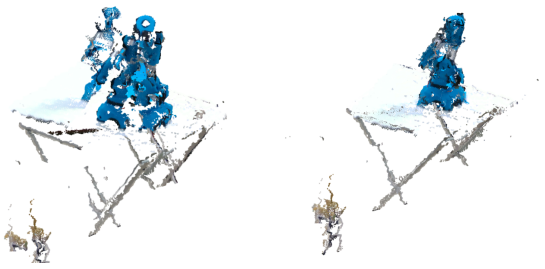
TABLE II: Distance errors Before vs After running GPA-ICP, for 3 different initializations.

Init.		C1-C2	C1-C3	C1-C4	C2-C3	C2-C4	C3-C4	Average
1	before	0.108	0.035	0.092	0.017	0.013	0.033	0.050
	after	0.023	0.011	0.019	0.036	0.006	0.014	0.018
2	before	0.012	0.016	0.052	0.084	0.062	0.056	0.047
	after	0.003	0.025	0.016	0.006	0.015	0.041	0.018
3	before	0.046	0.009	0.067	0.108	0.051	0.071	0.058
	after	0.008	0.022	0.012	0.005	0.003	0.042	0.015

By performing the GPA-ICP algorithm for each initialization, we obtain better poses, expressed by the overall drop in error after this step is performed as it can be seen through the average error row. This step is particularly important to refine poses that previously had high distance errors since we see all of those values drop below 2.3cm. However it seems this step improves the overall poses given from the initializations, at the cost of slightly increasing the error in one of the distances, which are C2-C3, C1-C3 and C1-C3 for initializations 1, 2 and 3 respectively, where camera 3 is always present. By analysing Fig. 11, we understand that camera 3 generates the point cloud that is most different, i.e. has less overlap, from the others, since it mostly observes the side of the robot, and for that it will be mostly neglected by GPA-ICP.

From this we conclude that, the method developed in this thesis to generate initializations manages to generate camera poses where cameras distances are miscalculated by roughly 5 centimeters on average as per Table II. Moreover its results depend on how the calibration object is moved around the scene during the calibration process. Regarding the GPA-ICP, we come to the understanding it that is yields accurate results when there is some overlap between camera views.

1) 3D Data Fusion: Having the poses, we can now merge the point clouds from every camera to see how accurate initialization and the GPA-ICP step are. By capturing a point cloud at the same time instant in each camera and applying to them the obtained camera poses from the initialization in Fig. 10b we can see a 3D reconstruction in Fig. 12a. By using refined poses Fig. 12b is obtained.



(a) (b)

Fig. 12: Point Cloud of Robot Arm on Table. (a) Before GPA-ICP (b) After GPA-ICP

From this we apprehend that initially obtained poses cause

some misalignment between point clouds. Moreover, the calculated poses have mostly an error in their translation, as rotation correctly matches. Using GPA-ICP correctly aligns the point clouds, and their respective camera poses.

VI. Application

To show the relevance of the method, it was utilized to improve face detection in Feedbot [3], a human feeding application. Nowadays there are many people with disabilities that cannot / have a lot of difficulty feeding themselves. The Feedbot system in Fig. 13 consists of an RGB-D camera and robot arm attached to a spoon. The camera is used to detect and track a person's mouth position and the robot arm is used to move food towards it. The developed method's contribution to this application is twofold. Firstly using the calibration multiple camera network, it is easier to track the user and everything around him to detect people, actions and any other kind of 3D information. Secondly the system can be used before hand to obtain a complete 3D face model, that can be easily application to the current Feedbot system. As so this second implementation was tested on the system.

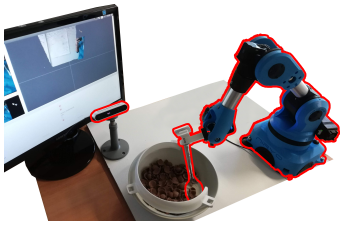


Fig. 13: Feedbot Feeding System.

The complete 3D face model from Fig. 14 was obtained using our method by placing a user in front of the cameras and cropping the points relative to his face.

(a) (b) (c)

Fig. 14: 3D Face Model. (a) Face Side View. (b) Face Top View. (c) Colored Face.

Having a more complete face model will be useful for tracking because Feedbot uses the Discriminative Optimization (DO) algorithm, which fit one point cloud to another by first having a training phase where it learns how to look for the 3D face model in a other point clouds. It then uses what it learned to track in real time the learned model.

Afterwards, the full 3D face model is used in Feedbot to check how well it helps the system find the face of the user face in real time. It can be seen in Fig. 15 that the face detection software based on DO correctly finds the face of the user and using facial landmarks, and his/her mouth, represented by the red sphere. The robot then uses this position to move the food filled spoon towards the users' mouth.

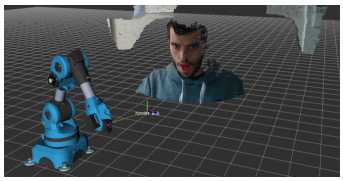


Fig. 15: Computer vizualisation of the Feedbot system.

VII. System Description and Functionalities

The system developed in this paper, was made to be easily scaled and altered, both hardware and software wise. It consists on a main computer where most of the computations occur, and five Intel RealSense ZR300[3] 3D cameras each one connected to an Upboard[4], a small form factor computer used to local process data and transmit it to the main server. The main server and the UpBoards are connected using a switch thus creating a local network, that used the ROS to communicate. The network diagram can be seen in Fig. 16.

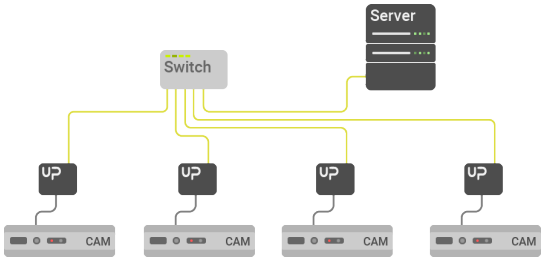


Fig. 16: System Diagram.

A. Functionalities

After the developed system is calibrated using the methods in III-D and IV-A, it can be used with two main purposes, that can have various applications.

Full Model 3D Reconstruction: By positioning the cameras around an object, it is possible the system to obtain a complete 3D model of it.

Real-Time 3D Reconstruction: By positioning cameras in an area, in such a away that the camera network calibration is not compromised, it is possible to obtain a Real-Time 3D reconstruction of everything in that area.

B. Limitations

In the current framework system most of the processing is done on the main server. This means that when we want to fetch the camera transformations, each individual camera image stream is sent to the server where the processing occurs. This may be problematic for systems with a larger number of cameras. Instead of sending image streams, an

[3]https://click.intel.com/media/ZR300-Product-Datasheet-Public-002.pdf
[4]https://up-board.org/up/specifications/

alternative would be to detect the calibration object in each individual UpBoard, and then send only a stream of the calibration object's pose, which would greatly reduce the network bandwidth usage.

Another design simplification was the use of rectified images, i.e. images that have no distortion. This means that images from cameras with some distortion (radial or tangential) must be undistorted beforehand.

C. Scalability and Flexibility

This current system can be easily scaled up as it is fairly simple to configure a new 3D camera + UpBoard pair, and add them to the network. Furthermore it is possible to use the system with other commercially available 3D cameras and also combinations of different cameras, as long as there is a ROS package made by the manufacturer to integrate them with the built framework. Finally, it is also possible to use just the calibration framework developed without using ROS nor hardware since the method can use previously obtained RGB images, instead of the real-time images from each camera.

VIII. Conclusions and Future Work

In this work we created a methodology to easily calibrate any network of RGB-D cameras. This system can be used to obtain a 3D Reconstruction of the surround space and or specific objects. This methodology comprises 2 steps. The first one consists in an initialization of the camera poses, while the second refines this results using a point cloud matching algorithm. This combination led to a very accurate camera registration.

We use existing metrics and created new ones to measure the accuracy of the results for every step of the progress, and reached the conclusion that the method works, is quick, accurate and it is easily expandable to include more, and different kinds of cameras.

An application setting was tested, Feedbot, a feeding a robot arm, to aid people with upper body mobility impairments to eat their meals. The applicability of the method was proved by the well-succeeded tracking of a face using our full 3D face model as a reference, thus allowing the robot to find the person's mouth.

In the future we wish to combat some of the current system limitations, so as to enable easier usage and scalability. This means making the system more distributed resulting in less information being sent through the network, make it more robust to camera desynchronization and more user-friendly so that anyone can use it.

Acknowledgment

The author would like to thank Professor Doctor Manuel Marques and Professor João Paulo Costeira for the guidance provided in this research.

References

[1] C. Chen, J. Favre, G. Kurillo, T. P. Andriacchi, R. Bajcsy, and R. Chellappa, "Camera networks for healthcare, teleimmersion, and surveillance," *Computer*, vol. 47, no. 5, pp. 26–36, May 2014.

[2] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2411–2418.

[3] A. Candeias, T. Rhodes, M. Marques, J. P. ao Costeira, and M. Veloso, "Vision augmented robot feeding," in *The European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[4] R. Toldo, A. Beinat, and F. Crosilla, "Global registration of multiple point clouds embedding the generalized procrustes analysis into an icp framework," in *3DPVT 2010 Conference*, vol. 2, 2010.

[5] "Vicon — award winning motion capture systems." [Online]. Available: https://www.vicon.com/hardware/cameras/

[6] "The Captury — Markerless Motion Capture technology." [Online]. Available: https://thecaptury.com/

[7] "Qualisys — motion capture systems." [Online]. Available: https://www.qualisys.com/hardware/

[8] P.-C. Su, J. Shen, W. Xu, S.-C. Cheung, and Y. Luo, "A fast and robust extrinsic calibration for rgb-d camera networks," *Sensors*, vol. 18, no. 1, p. 235, 2018.

[9] M. Ruan and D. Huber, "Calibration of 3d sensors using a spherical target," in *2014 2nd International Conference on 3D Vision*, vol. 1. IEEE, 2014, pp. 187–193.

[10] T. Svoboda, D. Martinec, and T. Pajdla, "A convenient multicamera self-calibration for virtual environments," *Presence: Teleoper. Virtual Environ.*, vol. 14, no. 4, pp. 407–422, Aug. 2005.

[11] X. Chen, J. Davis, and P. Slusallek, "Wide area camera calibration using virtual calibration objects," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, vol. 2, June 2000, pp. 520–527 vol.2.

[12] M. Munaro, F. Basso, and E. Menegatti, "Openptrack: Open source multi-camera calibration and people tracking for rgb-d camera networks," *Robotics and Autonomous Systems*, vol. 75, pp. 525–538, 2016.

[13] 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, 5 2016.

[14] A. Aalerud, J. Dybedal, and G. Hovland, "Automatic calibration of an industrial rgb-d camera network using retroreflective fiducial markers," *Sensors*, vol. 19, no. 7, p. 1561, 2019.

[15] M. Kowalski, J. Naruniec, and M. Daniluk, "Livescan3d: A fast and inexpensive 3d data acquisition system for multiple kinect v2 sensors," in *2015 International Conference on 3D Vision*, Oct 2015, pp. 318–325.

[16] CMU, "CMU Panoptic Dataset," 2015. [Online]. Available: http://domedb.perception.cs.cmu.edu/dataset.html

[17] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb 1992.

[18] G. D. Evangelidis, D. Kounades-Bastian, R. Horaud, and E. Z. Psarakis, "A Generative Model for the Joint Registration of Multiple Point Sets," in *European Conference on Computer Vision*, vol. 8695 LNCS. Springer, 2014, pp. 109–122.

[19] B. Eckart, K. Kim, and J. Kautz, "Fast and Accurate Point Cloud Registration using Trees of Gaussian Mixtures," *arXiv preprint arXiv:1807.02587*, 2018.

[20] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-d rigid body transformations: A comparison of four major algorithms," *Machine vision and applications*, vol. 9, no. 5-6, pp. 272–290, 1997.

[21] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision*, vol. 81, no. 2, p. 155, 2009.

[22] D. Akca, "Generalized procrustes analysis and its applications in photogrammetry," ETH Zurich, Tech. Rep., 2003.