Fast and Accurate Reconstruction of Pan-Tilt RGB-D Scans via Axis Bound Registration

Jung-Hyun Byun and Tack-Don Han

Abstract—A fast and accurate algorithm is presented for registering scans from an RGB-D camera on a pan-tilt platform. The pan-tilt RGB-D camera rotates and scans the entire scene in an automated fashion. The proposed algorithm exploits the movement of the camera that is bound by the two rotation axes of the servo motors so as to realize fast and accurate registration of acquired point clouds. The rotation parameters, including the rotation axes, pan-tilt transformations and the servo control mechanism, are calibrated beforehand. Subsequently, fast global registration can be performed during online operation with transformation matrices formed by the calibrated rotation axes and angles. In local registration, features are extracted and matched between two scenes. False-positive correspondences, whose distances to the rotation trajectories exceed a threshold, are rejected. Then, a more accurate registration can be achieved by minimizing the residual distances between corresponding points, while transformations are bound to the rotation axes. Finally, the preliminary alignment result is input to the iterative closed point algorithm to compute the final transformation. Results of comparative experiments validate that the proposed method outperforms state-of-the-art algorithms of various approaches based on camera calibration, global registration, and simultaneous-localization-and-mapping in terms of root-mean-square error and computation time.

Index Terms—point cloud registration; indoor scene reconstruction; multi-view alignment; rotation axis calibration; pan-tilt control

1 INTRODUCTION

Indoor scene reconstruction is of crucial importance in various research areas, in which 3D geometry data is required, such as robotics, architecture, and augmented reality. More specifically, in robotics, 3D map data is required for a robot to localize itself, plan a path, and navigate to a specific location as ordered. The practice of this localization and mapping is often abbreviated as SLAM (Simultaneous Localization and Mapping) [1], [2]. In architecture, indoor scenes are reconstructed, so that users can navigate virtually, and are then exported as CAD files for users to correct reconstruction errors and further edit to meet their needs [3]. In conjunction with augmented reality (AR), reconstructed models are used as graphics scene files, on which users can superimpose virtual objects before actually installing them [4], or are used as surface geometry for projection-based AR, where geometric distortions are corrected to deliver pure immersive AR experience [5], [6].

In general, RGB depth (RGB-D) cameras are hand-held by the user and are moved freely around the environment to map and reconstruct the geometry of indoor scenes, particularly for SLAM applications [7]. However, in various cases, RGB-D cameras are attached to pan–tilt rotating platforms and collect 3D data captured in pan–tilt sweeps [4], [5], [8], [9]. Using pan–tilt platforms with RGB-D cameras has several advantages. Namely, the field-of-view of the cameras is limited when a wide range of data is captured from the environment, and rotating the camera by pan–tilt units is preferred over noisy hand-held alternatives. Moreover, using pan–tilt RGB-D cameras in the center of a standard-sized room, is an ideal option for capturing room geometry [8] if the stationary nature of the scene and the ranging limit of the RGB-D camera (typically 1.5–4.5 m) are taken into account. Finally, as the scanning and capturing process can be automated and computer-controlled [4], [9], it would be more convenient and accurate to use pan–tilt RGB-D cameras.

As SLAM methods can perform localization of the camera that is freely moving and mapping of the environment at the same time, some pan–tilt setups adopts SLAM methods for such purposes. For example, Beamatron [5] is a steerable projection augmented reality system, where a projection and a Microsoft Kinect 360 were mounted on a pan-tilt moving light platform. Beamatron used the well-known KinectFusion [11] to reconstruct the surface geometry on which the projection was mapped. However, even though SLAM methods such as KinectFusion are able to reconstruct indoor scenes reliably, their typical reliance on the iterative closest point (ICP) algorithm [12] makes the overall process exhaustive and iterative. Thus, several ICP-based SLAM methods support real-time performance only with the assistance of massively parallel programming, namely general-purpose computing on graphics processing units (GPGPU) [13].

The prerequisite for GPU equipment of the SLAM approach sets a limitation for its applicability to low-powered computing devices that are without additional hardware. However, ICP-based SLAM methods would not be a necessity to track the camera pose or reconstruct the environment, if some prior knowledge of the camera motion is known, such as a pan–tilt platform. In this study, a fast and accurate registration algorithm, called axis bound registration, is proposed that can be used in such a configuration, where an RGB-D camera on a pan–tilt platform is rotated to
acquire scan data for reconstructing indoor environments. Exploiting the fact that the movement of the camera is bound by the two rotation axes of the servo motors, the proposed algorithm is able to produce comparable or better registration results in much shorter time, when compared to other state-of-the-art registration and SLAM methods.

The proposed axis bound registration is not technically a SLAM. However, the proposed method and SLAM methods have certain similarities: The proposed method models the motions of the camera (on a pan–tilt platform), estimates its pose changes (along the trajectory of rotations), and uses the acquired information to reconstruct the scene (by point cloud registration). Thus, the proposed method may be considered an alternative to SLAM methods when pan–tilt RGB-D camera systems are used, such as Beamatron or AIR. The results in Table 2 support the claim that the proposed method offers a lighter and more accurate option for scene reconstruction in pan–tilt cases. This is possible due to several novelties and contributions of the proposed axis bound registration, which are as follows:

1) The proposed method models the rotational motion of the pan–tilt platform, and thus it can estimate unseen poses of pan–tilt rotations. This is possible by prior knowledge acquired through modeling and recovering parameters of the rotation axes.

2) A novel rotation trajectory-based keypoint matching and rejection method is developed to further refine the estimated pose during online operation. This mitigates any negative effects of possible errors from the calibration process on pose estimation.

3) The alternating optimization removes most of the iterative nature of non-linear optimization in the pairwise registration process. This results in greatly improved registration accuracy/speed and convergence in the iterative closest point (ICP) process.

Overall, the proposed method outperforms state-of-the-art algorithms of various approaches based on camera calibration, global registration, and localization mapping in terms of root-mean-square-error (RMSE) and computation time.

In the following sections, a step-by-step approach to performing axis bound registration will be presented, which is the key algorithm of the proposed method. In Section 2, a review of related studies is presented. In Sections 3.1 and 3.2, the rotation axes of a pan–tilt platform are calibrated. In Section 3.3, the calibrated rotation parameters are then recovered in the global frame coordinates, so that all local point clouds may be registered in a common frame, using the global transformation model of Equation 8. In Section 4.1, the rotated angles of each local frame around the calibrated rotation axes are computed and the servo control mechanism that converts applied pulses to rotation angles is identified. Using the rotation parameters in the global frame and rotation angles, the current pose of the pan–tilt camera can be estimated and used to roughly register two point clouds globally in Section 4.2. In Section 4.3, the proposed method that further optimizes local registration results in a pairwise manner is explained using trajectory-constraint-based matching and rejection, and finally, the axis bound registration method is presented. In Section 5, the experiment design and configurations are described. Moreover, the performance of the proposed method against four other state-of-the-art registration methods is evaluated and validated.

2 RELATED WORK

2.1 Rotation Axis Calibration

As explained above, the proposed axis bound registration method extends pan–tilt axis calibration to achieve point cloud registration. It was inspired from various previous studies. The first to be reviewed are those from the computer vision field, where pan–tilt–zoom (PTZ) cameras have long been used for surveillance, object tracking and so forth. Davis and Chen in [14] presented a calibration model where the typical pin-hole camera model was extended to incorporate the characteristics of pan and tilt motions. Its most significant contribution is a pan–tilt assembly model that accurately reflects the motion and structure of a general pan–tilt camera, in which the pan and tilt axes are modeled independently from the axes of the camera frame. In [15], Wu and Radke proposed a camera model involving not only pan and tilt motions but also zoom on the image domain. They also reported their observations on the behavior of pan–tilt motors and categorized several errors that can arise during the operation of a PTZ camera.

In the optics and measurement literature, a calibration method for a rotating turntable was introduced in [16]. In the proposed setup, in contrast to a typical PTZ camera setup, the camera was fixed, whereas a checkerboard was attached to a turntable and rotated to acquire the rotation axis with respect to the fixed camera. Here, constrained global optimization and the least squares method were used to solve the great circle equations, which are used as models for the rotation trajectories. The method was later extended by Niu et al. in [10], where the camera itself was attached to a rotating plate. A regular RGB camera was used, and thus the camera could not directly see the rotation axis. Accordingly, another external camera was used as a global reference frame, and the rotation axis was computed with regard to the external camera.

Byun et al. in [17] proposed a control mechanism for manipulating the motion and orientation of a generalized pan–tilt camera. Combining rotation axis calibration and camera-servo control, the authors were able to control the camera and accurately target a designated mark, based on the inverse kinematics approach. Even though their method was accurate, it relied on iterative and time-consuming Jacobian optimization and did not incorporate random errors that can occur during operation, as reported in [15]. The proposed method significantly extends the method in [17] by incorporating a random servo error handling mechanism and registering RGB-D scans based on the pan–tilt camera motion and control model.

2.2 Point Cloud Matching

[18] defines the point registration as the process of acquiring a transformation between two point clouds that determines the mapping relationship in the 3D space. [19] and [18] categorized point cloud registrations into three types, namely, global, local, and local descriptor registration. Global registration is a method that globally searches
the point correspondences given two point clouds of any condition. By contrast, local registration methods require the input point clouds to be as close and overlapping as possible for the methods to work properly. Local descriptors registration involves a matching process of feature descriptors to determine point correspondences.

Super 4-points congruent sets (Super4PCS) [19] and fast global registration (FGR) [20] are examples of global registration methods. The authors of [19] proposed a novel method for removing the quadratic time complexity of its predecessor in [21]. The key idea behind this improvement is the use of the data structure in solving the core instance problem, where the goal is to find all candidate pairs of a given point that are within a distance range. FGR optimizes an objective function involving candidate matches over the surface of the object scans to align surfaces. The authors argue that the method does not require initialization, yet it can achieve accuracy comparable to that of well-initialized local refinement algorithms. Both global registration methods are compared with the proposed method in Section 5.

The proposed method adopts the coarse-to-fine registration scheme, where in the coarse registration step the two given RGB-D frames are brought to the same global coordinates, and in the fine registration step, the source frame is locally transformed and aligned to match the target frame. The global transform is achieved by modeling and recovering the rotation motions of the pan–tilt system. In the local transform, the keypoints are extracted and matched. False-matched keypoints are rejected by their distances to the rotation trajectory of the target points, and the remaining pairs are used to compute the transformation that brings corresponding points together, while keeping the transformations bound to the rotation trajectory. In this sense, the proposed method can be regarded as a hybrid global and local registration method, where the coarse registration step can be thought of as the global registration, and the fine registration step as the local registration.

### 2.3 Localization and Mapping

Research in the SLAM field focuses on positional, orientational tracking of the camera and reconstruction of the scene in tandem. Past studies on SLAM include MonoSLAM [22], which is based on the input of a single commodity sensor that is a monocular RGB camera, parallel tracking and mapping [23], which is more accurate, and DTAM [24], which combines camera tracking with dense surface reconstruction.

As mentioned above, some SLAM methods rely on the ICP algorithm [12] to track the pose of the camera. For example, [1] tracks the pose of the camera by repeatedly revising the transformation (translation and rotation) required for minimizing the difference between two clouds of points based on ICP. KinectFusion [11] and RGB-D SLAM [25] also depend on ICP to densely track the 3D pose of the Kinect sensor and reconstruct 3D surface models of the environment. In [26], ICP was used to add a graphical model layer that registers point clouds to optimize the pose graph and the odometry.

Even though the ICP algorithm is straightforward and functional, its exhaustive and iterative nature makes SLAM methods inherently heavyweight in terms of computing resources, such as power and memory consumption. Thus, several ICP-based SLAM methods is able to perform in real-time only with the assistance of GPGPU [13]. This includes [11], [25] or ElasticFusion [27], which all makes significant use of the GPGPU computation for mapping construction as well as pose reconstruction and tracking.

Recently, a novel ORB-SLAM2 method was proposed in [2] that accepts any camera type (stereo, RGB-D, or monocular) and creates synthetic stereo coordinates of extracted features. The authors explain that their method, based on bundle adjustment, achieves a globally consistent sparse reconstruction and is thus sufficiently lightweight to work with standard central processing units (CPU). However, even ORB-SLAM2 relies on CPU multi-threading, OpenMP in this case, to achieve real-time performance, and it is observed that the execution time degrades to one frame per second without multi-thread support (see results in Table 5).

### 3 Pan–Tilt Rotation Model

To accurately and efficiently register pan–tilt scans of an RGB-D camera, the proposed approach takes into account the fact that the camera movement on a pan–tilt platform can be estimated by modeling and calibrating the rotation parameters of the pan–tilt platform. To generalize the proposed method as much as possible, it is assumed that no detailed information regarding the pan–tilt system, such as its assembly configuration or factory calibration, is given. In this manner, the rotation motion of such generalized systems, the pan–tilt assembly model in [14] is adopted. The axes of pan and tilt rotations are not assumed to intersect the optical center of the camera, or to be precisely axis-aligned/orthogonal to each other. Pan and tilt motions modeled as rotations around arbitrary axes in space are shown in Figure 1. It should be noted that all transformations described hereafter are in a right-handed coordinate system, where the right, up, and front of the camera point in the X+, Y+, and Z− directions, respectively.

3.1 Pan–Tilt Calibration Process

To calibrate the axes of pan and tilt rotations, the method of Chen et al. [16] is employed, which was originally designed to calibrate the rotation axis of a turntable with a fixed camera. This is opposite to the aim of rotating the camera itself. However, as all transformations can be thought of as relative motions, one need only reverse the direction...
of the rotation, i.e., the sign of the angle, after all rotation parameters are recovered. As the exact rotation axes are not known to recover the pose in the global frame before the calibration, the pan and tilt axes are first calibrated in some reference frame. Then, the relative pose of the reference frame to the global frame is computed and the parameters are recovered in global coordinates later. The overall process for calibrating the rotation axis is shown in Figure 2.

First, an arbitrary reference frame is set as a base. Then, the system rotates itself by a certain step using some units (in the present case, pulse width modulation or PWM is used). While rotating, the camera captures the coordinates of the checkerboard corners and saves them after converting them into depth camera coordinates, with the pulse width applied to the servo motor. These steps are iterated after the rotation is reversed until the checkerboard is out of view in the captured image, at both ends of the field-of-view, collecting as many frames of the checkerboard as possible. Based on the checkerboard normal, the reference pose in the global frame is later estimated.

Rotating arbitrary points around a rotation axis forms closed, circular trajectories, which can be represented as great circles on 3-dimensional planes, with the plane normals being rotation axes and the circle centers representing pivot points. Let $l$ denote the count of the frames captured throughout the calibration process. Then, there exist in total $l$ instances of a certain point, one from each captured frame. These $l$ coordinates from multi-view frames can be used to obtain the circular trajectory of this certain point using the global least squares method [28], as in [16], [17]. Here, arbitrary points are corners on the checkerboard plane, with their geometrical relationship pre-known. Thus, all rotation trajectories are not independent, and can be represented relative to that of the top-leftmost corner. Then, the model can be parameterized from all corners of all frames, and the objective function can be globally optimized. Assuming that the checkerboard comprises $m$ rows and $n$ columns of corners, $l \times m \times n$ corners in total can be used in the global optimization.

Let the rotation direction vector be defined as $n = [n_x, n_y, n_z]^\top$, with its norm being $||n|| = 1$, and the rotation center for the upper-left corner be defined as $p = [a, b, c]^\top$. Then the line equation defining its rotation axis in 3D space is

$$\frac{x-a}{n_x} = \frac{y-b}{n_y} = \frac{z-c}{n_z}. \quad (1)$$

Furthermore, let the plane on which the circular trajectory of the rotation lies and that is $d$ millimeter away from the origin be defined as

$$n_x x + n_y y + n_z z + d = 0. \quad (2)$$

The rotation circles of checkerboard corners have the same rotation axis, and the corners have the same geometric structure as the checkerboard. Therefore, their rotation centers, which are defined on the same line by Equation 1, can be represented using the interval terms in conjunction with the row and column indices. Let the distances of the intervals between each corner plane be denoted by $d_h$ and $d_w$ (vertically and horizontally), respectively. Then, the rotation circle center of an arbitrary corner on the $i$-th row and $j$-th column of the checkerboard can be denoted by $p_{ij} = [a_{ij}, b_{ij}, c_{ij}]^\top$, where

$$a_{ij} = a - n_x (id_h + jd_w),$$
$$b_{ij} = b - n_y (id_h + jd_w),$$
$$c_{ij} = c - n_z (id_h + jd_w). \quad (3)$$

The proposed axis bound registration method requires calibration and modeling of the rotational motions of the pan and tilt servo motors. The steps in [17] are followed to recover their parameters for the rotation model, which is shown in Figure 3. The calibration process is divided into two sub-processes, where the direction vector of the rotation is first estimated by plane fitting, and the rotation trajectory is estimated by circle fitting. During the calibration, a large checkerboard is placed in front of the camera, vertical to the ground plane. Then, the camera is rotated clockwise and counter-clockwise until the checkerboard is out of view in the captured image, at both ends of the field-of-view, collecting as many frames of the checkerboard as possible.

3.2 Rotation Axis Calibration
The rotation trajectory is ideally modeled as a great circle in 3D space, which is uniquely and jointly defined by a plane and a sphere intersecting the plane. Here, the plane is modeled with parameters as

\[ n_x x + n_y y + n_z z + d + id_h + j d_w = 0, \]  

and the sphere as

\[ (x - a_{ij})^2 + (y - b_{ij})^2 + (z - c_{ij})^2 = r_{ij}^2, \]

where \( l \) is the count of captured frames, \( m \) is the count of checkerboard rows, and \( n \) is the count of columns corners, as defined above. Let \( [x_{ijk}, y_{ijk}, z_{ijk}]^T \) denote the 3D coordinates of the checkerboard corner on the \( i \)-th row and \( j \)-th column in the \( k \)-th captured frame. Then, a cost function is constructed to compute the parameters \( n_x, n_y, n_z, d, d_h, d_w \) minimizing the error of violating the plane model constraint, as follows:

\[ \sum_{k=0}^{l-1} \sum_{m=0}^{m-1} \sum_{n=0}^{n-1} (n_x x_{ijk} + n_y y_{ijk} + n_z z_{ijk} + d + i d_h + j d_w)^2. \]

With parameters recovered from Equation 3 and their relationship in Equation 6, another cost function is constructed to identify the values of the parameters \( a, b, c, r_{ij} \) minimizing the error of violating the circle model constraint, as follows:

\[ \sum_{k=0}^{l-1} \sum_{m=0}^{m-1} \sum_{n=0}^{n-1} ((x_{ijk} - a_{ij})^2 + (y_{ijk} - b_{ij})^2 + (z_{ijk} - c_{ij})^2 - r_{ij}^2)^2. \]

All \( l \times m \times n \) coordinates of checkerboard corners in multi-view frames are used to minimize two cost functions. The optimization is carried out using the global least squares method, which determines the optimal values for the rotation parameters [16].

### 3.3 Pan–Tilt Transformation Model

In Section 3.2, the parameters for modeling the rotations, namely, their axes and pivot points, were calibrated. With all the parameters of pan and tilt rotations calibrated, the corresponding coordinates of 3D points between a local camera frame and the reference frame can now be represented. If the pan and tilt angles of the local frame are denoted by \( \alpha \) and \( \beta \) respectively, and the local point by \( P_{local} \), its corresponding point in the reference frame can be modeled as follows [17):

\[ P_{ref} = T_{pan} R_{pan}(\alpha) T_{tilt} R_{tilt}(\beta) T_{tilt}^{-1} P_{local}^{-1}. \]

where

\[ R(\theta) = \begin{bmatrix} \cos \theta + n_x^2 (1 - \cos \theta) & -n_x n_y (1 - \cos \theta) & n_x n_z (1 - \cos \theta) + n_y \sin \theta \\ -n_x n_y (1 - \cos \theta) & \cos \theta + n_y^2 (1 - \cos \theta) & -n_y n_z (1 - \cos \theta) - n_x \sin \theta \\ n_x n_z (1 - \cos \theta) - n_y \sin \theta & n_y n_z (1 - \cos \theta) + n_x \sin \theta & \cos \theta + n_z^2 (1 - \cos \theta) \end{bmatrix}, \]

and

\[ T = \begin{bmatrix} 1 & 0 & 0 & d_h \\ 0 & 1 & 0 & d_w \\ 0 & 0 & 1 & d_w \\ 0 & 0 & 0 & 1 \end{bmatrix}. \] (8)

Here, \( R_{tilt} \) is a \( 4 \times 4 \) matrix representing the tilt rotation around its axis vector \( n = [n_x, n_y, n_z]^T \) and \( T_{tilt} \) is a \( 4 \times 4 \) matrix that represents the position of the rotation axis \( c = (c_x, c_y, c_z) \) that uniquely defines the tilt rotation.

Any point on the line of the rotation axis can be a pivot for the rotation, which does not affect the final transformation. Here, the center point of the rotation of the upper-leftmost corner of the checkerboard was used. The same representation applies to the pan transformation.

It should be noted that the modeling was performed so that the system first tilts and then pans. This is to reflect the kinematic properties of the actual pan–tilt system that was used. As shown in Figure 8, the tilting arm is attached to the platform that is rotated by the panning servomotor. Thus, by applying the tilt transformation first and then the pan transformation, their rotations can be uniquely represented in the global system.

### 3.4 Rotation Parameterization in Global Frame

The parameters obtained in Section 3.2 however, are in the arbitrary reference coordinate system. This implies that all transformations using Equation 8 are in the reference frame. As the reference frame is arbitrarily chosen in Section 3.1, these parameters, two axes \( n_{pan}, n_{tilt} \) and two pivots \( c_{pan}, c_{tilt} \), should be recovered in the global frame. The global frame is defined as the coordinate system of the RGB-D camera when it is directly facing the checkerboard and is perpendicular to the ground plane. The angles of pan and tilt rotations are defined as zero when the camera is in such a pose, and are denoted by \( \theta_{pan} \) and \( \theta_{tilt} \) when the camera is in the reference frame. The \( \theta_{tilt} \) and \( \theta_{pan} \) angles of the reference frame can be calculated based on the normal vector of the checkerboard.

To calculate the normal vector of the checkerboard plane, principal component analysis (PCA) is performed on the checkerboard corners. The corners are first translated so that their mean is equal to zero. Then, the covariance matrix of the checkerboard corners is considered. Decomposing the covariance matrix into eigenvectors and using the corresponding eigenvalues yields the estimate of the plane orthogonal basis. The normalized eigenvector corresponding to the minimum eigenvalue is the minimum mean square error solution to the plane normal estimation. The above covariance matrix \( Q \) can be expressed as follows:

\[ Q = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \begin{bmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{bmatrix} \begin{bmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{bmatrix}, \]

\[ = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \begin{bmatrix} (x_{ij} - \bar{x})^2 \\ (y_{ij} - \bar{y})^2 \\ (z_{ij} - \bar{z})^2 \end{bmatrix} \begin{bmatrix} (x_{ij} - \bar{x})(y_{ij} - \bar{y}) \\ (x_{ij} - \bar{x})(z_{ij} - \bar{z}) \\ (y_{ij} - \bar{y})(z_{ij} - \bar{z}) \end{bmatrix}. \] (9)

Comparing the checkerboard normal \( n_{board} \) with its projection onto the \( XZ \)-plane \( n_{proj} \) yields the \( \theta_{tilt} \) angle, whereas comparing the projection with the optical axis, or the \( Z \)-axis, yields the \( \theta_{pan} \) angle by the following equation:

\[ \theta_{tilt} = \arccos(n_{proj} \cdot n_{board} / |n_{proj}| |n_{board}|), \]

\[ \theta_{pan} = \arccos(n_{proj} \cdot [0, 1, 0]^T / |n_{proj}| |1|). \] (10)

These angles are substituted into Equation 8. Parts of the equation are rewritten to shorten it:
F_{tilt}(\theta_{tilt}) = T_{tilt} R_{tilt}(\theta_{tilt}) T_{tilt}^{-1} and \ F_{pan}(\theta_{pan}) = T_{pan} R_{pan}(\theta_{pan}) T_{pan}^{-1}. Then the relation equation becomes
\[ \begin{bmatrix} P_{global} \\ 1 \end{bmatrix} = F_{pan}(\theta_{pan}) F_{tilt}(\theta_{tilt}) \begin{bmatrix} P_{ref} \\ 1 \end{bmatrix}. \] (11)

As noted in Section 3.3, tilting precedes panning. This implies that the tilt parameters calibrated in Section 3.2 are not affected by the initial \( \theta_{pan} \) rotation, or \( F_{pan}(\theta_{pan}) \). That is, the calibrated \( n_{tilt} \) and \( c_{tilt} \) are actually in the global frame, and thus no additional transformation is required. By contrast, the pan parameters are computed after the initial \( \theta_{tilt} \) transformation has been applied. Therefore, in the case of pan, \( F_{tilt}(\theta_{tilt}) \) should be applied to the pivot translation to recover its position in the global frame. The pan axis, unlike its pivot, is a vector, not a point. Thus, only the rotation, without translation, should be applied. The above transformation can be summarized as follows:
\[ \begin{bmatrix} \hat{n}_{pan|global} \\ 1 \end{bmatrix} = R_{tilt}(\theta_{tilt}) \begin{bmatrix} \hat{n}_{pan|ref} \\ 1 \end{bmatrix} \] and
\[ \begin{bmatrix} c_{pan|global} \\ 1 \end{bmatrix} = T_{tilt} R_{tilt}(\theta_{tilt}) T_{tilt}^{-1} \begin{bmatrix} c_{pan|ref} \\ 1 \end{bmatrix}. \] (12)

The final transformation equation of a point to global coordinates is not different from Equation 8, except that as the transformation is now from the reference frame to the global frame instead of the local and reference, the global parameters recovered above should be used and the subscripts should change accordingly.

4 Pan–Tilt Axis Bound Registration

In this section, we explain the pan–tilt axis bound registration, which is the key approach of the proposed method. It should be noted that the proposed method is inspired by \[18\] and \[17\]. However, it distinguishes from and extends the original approaches. A brief overview, including motivation, contributions, and limitations, of these two studies is first presented.

In \[18\], an indoor scene point cloud registration method is proposed, where a pan–tilt RGB-D camera is used to capture and register point clouds. Multi-view calibrations are conducted to obtain the extrinsic parameter matrices with regard to the initial world coordinates prior to the point cloud registration process. The transformation matrices are stored and used as rough estimations for point cloud registrations during online operation. As the camera calibration from multiple viewpoints is performed offline using a known-geometry object, such as a checkerboard, the computation time for the transformation estimation is greatly reduced and the alignment precision is increased. However, as stated in \[18\], the method suffers from errors propagated from the calibration process, which leads to inaccurate point cloud alignments. Furthermore, the method cannot estimate the poses of the camera if the pan–tilt platform rotates to any unseen angles, as the transformation matrices should be calculated beforehand.

To tackle the issues in \[18\], the proposed \textit{axis bound registration} method is based on the pan–tilt camera control method of \[17\]. The authors proposed a mechanism for accurate control of a pan–tilt system with an RGB-D camera. They formulated pan–tilt rotations as motions along great circle trajectories and calibrated their model parameters in 3D space, such as positions and vectors of rotation axes. The original study \[17\] focused on accurately controlling the pose of the camera, namely, orienting to and targeting at a designated mark, by optimizing the rotation angles of its transformation using inverse kinematics. The Jacobian transpose method. The present study branches out from the camera control method into online pan–tilt pose estimation and the axis bound registration, using a different alternating optimization method.

4.1 Camera Transformation with Servo Control

The proposed method aims at fast and accurate registration of point clouds of pan–tilt scans. The key motivation for this approach is that a rough estimate of the camera’s pose in the global system can be obtained based on the servo control. The rotation parameters in the global frame and the initial pose of the reference are recovered in Section 3.4. The effect of servo control on the rotation angle of the pan–tilt platform should now be considered.

Servo motors use potentiometers to orient themselves to certain directions. When a pulse of a certain width is applied, the servo rotates a by certain portion of its rotation range linearly with respect to its duty cycle. Therefore, it is possible to estimate the rotated angle of the pan–tilt platform based on the width of the applied pulse by the following relationship:
\[ \text{pulse width} = \text{scale} \times \text{angular degree} + \text{offset} \] (13)

As the above equation is a linear regression model, the solutions for \textit{scale} and \textit{offset} can be conveniently obtained by minimizing least square errors when an adequate amount of paired \{pulse width, angular degree\} data is given. During the rotation axis calibration process, the pulse widths applied to \textit{pan} and \textit{tilt} servos to orient the RGB-D camera are also saved to be input to Equation 13. Based on Equation 8, one can calculate the angles of each \textit{local} frame rotated from the reference frame using \( m \times n \) corner points of the checkerboard.

In Section 3.4, the initial pose of the reference frame, \( \theta_{pan} \) and \( \theta_{tilt} \) was acquired. Substituting the values of \( \theta_{pan} \) or \( \theta_{tilt} \) into Equation 8 yields transformations that are solely based on \textit{pan} or \textit{tilt} rotations. Because \textit{pan} and \textit{tilt} rotations are separately calibrated, as mentioned in Section 3.4, these transformations can be used to compute the angular degrees by which each captured \textit{local} camera frames rotated during the rotation axis calibration process. For simplicity, only the case of \textit{pan} rotation is considered. Substituting \( \theta_{tilt} \) into Equation 8 yields the following equation:
\[ \begin{bmatrix} P_{global} \\ 1 \end{bmatrix} = T_{pan} R_{pan}(\alpha) T_{pan}^{-1} \begin{bmatrix} P_{local} \\ 1 \end{bmatrix}, \]
\[ \text{where} \begin{bmatrix} P_{local} \\ 1 \end{bmatrix} = T_{tilt} R_{tilt}(\theta_{tilt}) T_{tilt}^{-1} \begin{bmatrix} P_{local} \\ 1 \end{bmatrix}. \] (14)

Here, \( P_{global} = [x_g, y_g, z_g]^T \) are the coordinates of \( m \times n \) checkerboard corners of the reference frame represented in the \textit{global} system. \( P_{local} = [x_l, y_l, z_l]^T \) are the coordinates of checkerboard corners of each \textit{local} frame, with \( P_{local} = [x_l', y_l', z_l']^T \) being its \textit{tilted} form. Only \( \alpha \), which represents the rotation angle around the \textit{pan} axis, is an
unknown variable in the form of \( \sin \) and \( \cos \) in the equation. Simplifying Equation 14 with respect to \( \sin(\alpha) \) and \( \cos(\alpha) \) yields

\[
A(\theta_{\text{tilt}}) \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \\ 1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ z_2 \end{bmatrix}, \quad \text{where } A(\theta_{\text{tilt}}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.
\]

This may be resolved by setting the pulse width of the reference frame as the offset and solve for \( scale \) only with the differences, not the original values, of the pulse widths relative to each local frame. This brings down the order of magnitude difference to 0–1 and yields a more stable solution. After obtaining the solution, the offset is adjusted by the value of the initial pose angle, \( \theta_{\text{pan}} \) or \( \theta_{\text{tilt}} \), multiplied by \( scale \).

### 4.2 Global Registration

The motivation in this study is similar to that in [18] in the sense that advantage is taken of the pan–tilt system to estimate the current pose of the camera. The key difference is that the proposed method models the movement of the pan–tilt system, whereas [18] relies on a pre-constructed database. In [18], multi-view calibration was performed to compute the transformation matrices of each preset pan–tilt rotation. Then a database was constructed that stores all the matrices. Even though this approach saves computation time in the registration task, the method is limited in terms of scalability, as it is virtually impossible to record transformations of every possible pan–tilt point in the real world. To tackle this issue, the rotation of a pan–tilt system is modeled and estimated, instead of creating and looking up a database.

The pan–tilt system is rotated by two servo motors, which are controlled according to the width of the applied pulse. Thus, by formulating the rotation angles of the servo motors and their transformation model, the overall transformation of the pan–tilt system can be identified. The rotation degree can be estimated by inputting the pulse width to Equation 14. The rotation angle is then substituted into Equation 15 yielding the final transformation matrix. The transformation matrix is an estimation of the camera pose in the global frame, which can be interpreted as the \( \text{view}^{-1} \) matrix in graphics terms. Here, point clouds are captured in local camera coordinates, implying that their model matrices are actually the \( \text{view}^{-1} \) matrix of the camera. Thus, the camera pose in the global frame, which is used to convert local coordinates of 3D points in the camera frame to global coordinates [17], can be also used as a rough estimation for registering point clouds.
4.3 Local Registration

Using the RANSAC method to remove outliers is a common approach. However, the iterative nature of random sampling inevitably results in extensive computational load for reaching consensus. Thus, a novel method is introduced whereby the geometric relationships obtained in Section 4.1 are used as trajectory constraints to reject false positives. The overall process and schematic diagram for determining and rejecting outliers are shown in Figure 5. Let the first and second frames of a given pair be denoted by \( l \) and \( r \), respectively. A frame of pan–tilt scan data consists of a color (RGB) image, a depth image, and a point cloud. The objective is to align two point clouds \( P_l^k \) and \( P_r^l \). First, ORB keypoints are extracted from two color images \( C_l \) and \( C_r \), and are matched based on their descriptors. These keypoints in color image domains are then unprojected to 3D points in each local camera frame using depth data, \( D_l^k \) and \( D_r^l \). The two keypoints of a 3D keypoint pair are denoted by \( k_l^i \) and \( k_r^i \), where the subscript \( i \) denotes the \( i \)-th keypoint pair. The pulse widths of each frame can be converted to pan and tilt angles in global space using Equation 13, \( \alpha_l \), \( \beta_l \), \( \alpha_r \), and \( \beta_r \).

Substituting \( k_l^i \), \( \alpha_l \), \( \beta_l \), \( \alpha_r \), and \( \beta_r \) into Equation 13 yields 3D points with global coordinates, which are denoted by \( \tilde{k}_l^i \) and \( \tilde{k}_r^i \). The corresponding keypoint \( k_r^i \) of the \( r \) frame is obtained as follows:

\[
\begin{array}{c}
\hat{k}_l^1 = T_{\text{pan}} R_{\text{pan}}(\alpha_l) T_{\text{tilt}} R_{\text{tilt}}(\beta_l) T_{\text{tilt}}^{-1} \hat{k}_l^1 , \\
\hat{k}_r^1 = T_{\text{pan}} R_{\text{pan}}(\alpha_r) T_{\text{tilt}} R_{\text{tilt}}(\beta_r) T_{\text{tilt}}^{-1} \hat{k}_r^1 .
\end{array}
\]

(17)

Given \( \hat{k}_l^1 \) and \( \hat{k}_r^1 \), which are coordinates computed by initial estimations of the rotation angles, i.e., \( \alpha_l \), \( \beta_l \), \( \alpha_r \), and \( \beta_r \), it is determined whether they are falsely matched or not based on their distances. Specifically, the indicator function \( g(\hat{k}_l^1, \hat{k}_r^1) \) declares that the matching of \( \hat{k}_l^1 \) and \( \hat{k}_r^1 \) is a false positive if the distance exceeds a certain limit, namely \( \delta \), as follows:

\[
g(\hat{k}_l^1, \hat{k}_r^1) = \begin{cases} 
0, & \text{if } \|\hat{k}_l^1 - \hat{k}_r^1\|_2 > \delta \\
1, & \text{otherwise}
\end{cases}
\]

Here, \( \| \cdot \|_2 \) denotes the Euclidean distance function for the given input. As mentioned above, however, a pan–tilt system is inclined to introduce random errors in its rotation. Thus, even though the mapping between pulse width and rotation angle has been modeled by Equation 13, possible errors should be taken into account in the calculation of rotation angles. To mitigate the effect of the random errors on the indicator function \( g(\cdot, \cdot) \), the threshold value of the distance \( \delta \) is defined with sufficient margins to prevent false-negative rejections. In the proposed setup used in Section 5, \( \delta \) was set to 100 mm. The feature matching and rejection result is illustrated in Figure 7.

4.3.2 Pairwise Axis Bound Transform Estimation

For clarity, the notations \( \hat{k}_l^i \) and \( \hat{k}_r^i \) are used to denote keypoint pairs that are determined as true-positives by Equation 13. It should be noted that in pairwise registration, the \( l \) frame is the target frame and \( r \) is the query. The target frame can be thought of as the reference frame and the query as the local frame in Equation 13. Then, the objective of local registration is to determine the pan–tilt angles of \( \alpha_l^* \) and \( \beta_l^* \) of the query frame that transforms \( \hat{k}_r^i \) to \( \hat{k}_l^i \) using the inverse homography method to validate whether the randomly sampled points were in accordance with the target image in a geometrical sense.
Fig. 6: Schematic diagram for rejecting outlier matching pairs based on trajectory constraints.

(a) Keypoint pairs between two proximate frames are matched in the 2D image domain.
(b) Keypoints are unprojected to 3D points in global space, represented with halos.
(c) Keypoints whose distance is below the threshold $\delta$ are accepted as a correspondence pair.
(d) Keypoints whose distance is above the threshold $\delta$ are rejected as outlier matches.

$\text{arg min}_{\alpha^r, \beta^r} \sum_i N \left\| \hat{k}_i^l - T_{\text{pan}} R_{\text{pan}}(\alpha^r) T_{\text{tilt}} R_{\text{tilt}}(\beta^r) T_{\text{tilt}}^{-1} \left[ \hat{k}_i^r \right] \right\|^2,$

(19)

where $N$ is the number of matched keypoint pairs between frame $l$ and $r$.

Obviously, the cost function in Equation 19 is a non-linear optimization problem about two variables $\alpha^r$ and $\beta^r$, which are in the form of interdependent trigonometric functions $\cos$ and $\sin$. It is a common practice to solve a non-linear optimization problem in an iterative manner, reducing numerical errors step-by-step. Taking [17] for example, the authors solved a similar problem, where the objective was to orient the camera so that it positions...
Fig. 7: Illustration of (a) matching keypoints, (b) rejecting outliers, and (c) final correspondence pairs. Outliers are rejected if the distance in global coordinates to estimated pairs exceeds a certain threshold (e.g., 100 mm). Keypoint pairs with diagonal directions and weak correspondences are removed.

The target object at the center of the image. Adopting the inverse kinematics approach, they used the Jacobian transpose method [32] to iteratively minimize point-to-point distances. Likewise, relying on the inverse kinematics approach and iterative solving for two rotation angles in the proposed methods is a viable option. However, as iterative methods often hinder real-time performance, a more sophisticated scheme of alternating optimization is adopted, as was used in camera calibration implementation in the OpenCV library [33] (p.673), and camera pose optimization [34] (Section 4.3 Alternating Optimization). The alternating optimization scheme decomposes Equation 19 into two sub-problems, where \( \alpha^r \) is independently obtained with fixed \( \beta^r \) and vice versa. This continues iteratively until the system reaches numerical convergence. Even though the alternating optimization scheme itself does not solve the equation in a closed form, the sub-divided equations regarding \( \alpha^r \) and \( \beta^r \) can be solved in closed form, greatly reducing computational complexity. Moreover, it is empirically confirmed that the alternating optimization scheme solves Equation 19 in a semi-closed form, that is, a sufficiently optimal local registration solution is reached starting from a coarse global registration in a single step. The RMSE performance of the single-step alternating optimization scheme is discussed in Section 5.

A part of the alternating optimization scheme is already introduced in Equations 14 and 15. The fundamental ideas are the same. The difference is that the rotation angles for tilt are set according to Equation 13, not from the values obtained by the rotation axis calibration process. This is also the key difference from [18], where only rotation angles that are pre-acquired in the camera calibration process are stored and later used. Another difference is that matched keypoints are now used as local–global correspondence pairs, instead of checkerboard corners, thus compensating for random errors due to the servo motors [19] and further optimizing for better online local registration results.

Let \( \theta_{\text{tilt}} \) denote the given value for the tilt rotation. Then ideally, by the rotation transformation of Equation 8, the equation for the pan rotation angle \( \alpha \) can be modeled as follows:

\[
\begin{bmatrix}
\hat{k}_1^r \\
1
\end{bmatrix} = T_{\text{pan}} R_{\text{pan}}(\alpha) T_{\tilde{\text{pan}}}^{-1} \begin{bmatrix}
k_1^r \\
1
\end{bmatrix},
\]

where

\[
\begin{bmatrix}
k_1^r \\
1
\end{bmatrix} = T_{\text{tilt}} R_{\text{tilt}}(\theta_{\text{tilt}}) T_{\tilde{\text{tilt}}}^{-1} \begin{bmatrix}
k_1^r \\
1
\end{bmatrix}.
\]

Similarly, let \( \theta_{\text{pan}} \) denote the given value for the pan rotation. Then ideally, by the rotation transformation of Equation 8, the equation for the tilt rotation angle \( \beta \) can be modeled as follows:

\[
\begin{bmatrix}
k_1^l \\
1
\end{bmatrix} = T_{\text{tilt}} R_{\text{tilt}}(\beta) T_{\tilde{\text{tilt}}}^{-1} \begin{bmatrix}
k_1^l \\
1
\end{bmatrix},
\]

where

\[
\begin{bmatrix}
k_1^l \\
1
\end{bmatrix} = T_{\text{pan}} R_{\text{pan}}^{-1}(\theta_{\text{pan}}) T_{\text{pan}} \begin{bmatrix}
k_1^l \\
1
\end{bmatrix}.
\]

The \( r \) superscripts of \( \alpha \) and \( \beta \) are omitted for simplicity. \( \hat{k}_1^l, \hat{k}_1^r, k_1^l, k_1^r \) are all 1-by-3 column vectors denoting XYZ coordinates in the 3D space. It should be noted that \( \hat{k}_1^l \) and \( k_1^r \) are introduced to incorporate known rotations of \( \theta_{\text{tilt}} \) and \( \theta_{\text{pan}} \), respectively, for clarity. The cost function in Equation 20 can now be divided into two sub-cost functions as follows:

\[
\arg\min_{\alpha} \sum_i^N \left\| \begin{bmatrix}
k_i^l \\
1
\end{bmatrix} - T_{\text{pan}} R_{\text{pan}}(\alpha) T_{\tilde{\text{pan}}}^{-1} \begin{bmatrix}
k_i^r \\
1
\end{bmatrix}\right\|^2_2
\]

and

\[
\arg\min_{\beta} \sum_i^N \left\| \begin{bmatrix}
k_i^l \\
1
\end{bmatrix} - T_{\text{tilt}} R_{\text{tilt}}(\beta) T_{\tilde{\text{tilt}}}^{-1} \begin{bmatrix}
k_i^r \\
1
\end{bmatrix}\right\|^2_2.
\]

Each sub-cost function can be alternatingly solved in closed form. The alternating optimization starts by substituting the initially estimated tilt rotation angle of Equation 13 into Equation 22 as \( \theta_{\text{tilt}} \). Solving Equation 22 yields the value for \( \alpha \), which is substituted into Equation 23 as \( \theta_{\text{pan}} \) and fixed. Solving Equation 23 yields the value of \( \beta \). \( \beta \) can be in turn substituted into Equation 22 as \( \theta_{\text{tilt}} \), which can be solved again for further optimization. Empirically, a single step suffices to yield numerically optimal solutions. To solve Equations 22 and 23 two linear systems similar to
Equation [15] are constructed.

\[
A(\theta_{pan}) = \begin{bmatrix}
\cos(\beta) \\
\sin(\beta)
\end{bmatrix} = \begin{bmatrix}
k_i^t \\
1
\end{bmatrix}, \text{ where } A(\theta_{pan}) = \begin{bmatrix}
1 & 0 & 0 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
1 & 0 & 0
\end{bmatrix}
\]

(1) \(x_{i}' - x_{i}'' = (n_{x} - 1) x_{i}'' + n_{x} x_{i}' - n_{x} x_{i}''
\)

(2) \(y_{i}' - y_{i}'' = (n_{y} - 1) y_{i}'' + n_{y} y_{i}' - n_{y} y_{i}''
\)

(3) \(z_{i}' - z_{i}'' = (n_{z} - 1) z_{i}'' + n_{z} z_{i}' - n_{z} z_{i}''
\)

(4) \(x_{i}' - x_{i}'' = (n_{x} - 1) x_{i}'' + n_{x} x_{i}' - n_{x} x_{i}''
\)

(5) \(y_{i}' - y_{i}'' = (n_{y} - 1) y_{i}'' + n_{y} y_{i}' - n_{y} y_{i}''
\)

(6) \(z_{i}' - z_{i}'' = (n_{z} - 1) z_{i}'' + n_{z} z_{i}' - n_{z} z_{i}''
\)

Here, \(k_i^t = [x_i^r, y_i^r, z_i^r]^T\) and \(k_i^t' = [x_i'^{r}, y_i'^{r}, z_i'^{r}]^T\). In both Equations 24 and 25, concatenating both sides column-wise for \(N\) keypoint pairs results in two over-determined linear systems. The systems are classic examples that can be solved by the SVD method, resulting in least squares error solutions for \(\cos(\beta), \sin(\beta), \cos(\alpha), \sin(\alpha)\). The initial solutions are refined to obtain the actual rotation angles of interest by enforcing trigonometric properties, that is, \(\beta = \arctan(2(\sin(\beta), \cos(\beta)))\) and \(\alpha = \arctan(2(\sin(\alpha), \cos(\alpha)))\). The \(\alpha\) and \(\beta\) define the pairwise transformation from frame \(r\) to \(l\) in global space, which finalizes local registration between two frames.

\section{5 Experiment and Evaluation}

\subsection{5.1 System and Dataset Configuration}

To validate the proposed method, an experimental environment was set up comprising an RGB-D camera and two servo motors that rotate in \(pan\) and \(tilt\) directions, supported by a steerable platform, with an external large-sized checkerboard for calibration purposes. The checkerboard consists of \(8 \times 10\) black-and-white checkers, which implies that there are seven rows and ten columns of corner points. The size of a checker, or the distance between corners, is 100 mm vertically and horizontally. The steerable platform was assembled from custom-made arms and gears, where two servos were attached to rotate the platform horizontally and vertically. Two HS-785HB servo motors were used, which accept 600–2400 microseconds pulse width. The model is a so-called \textit{hobby servo}, which costs approximately 50.00 USD and can rotate with an acceptable accuracy; however, it is not industry-grade. For the RGB-D camera, Microsoft Kinect v2 was used, supporting up to 1920 \(\times\) 1080 RGB stream and 512 \(\times\) 424 time-of-flight depth stream in real time. It was mounted on the upper plate of the steerable platform. The setup used in the pan–tilt scan registration is shown in Figure 8.

The proposed approach, similar to that in [4], combines servo control and camera pose estimation for point cloud registration. Thus, calibration data, including rotation axes and pulse width mapping, is required as well as color images, depth images, and point cloud data. However, to the best of the authors knowledge, there is no publicly available dataset that meets all the requirements for pan–tilt RGB-D scan registration, including the dataset in [3]. Therefore, the pan–tilt RGB-D scan dataset was constructed in-house, using the setup shown in Figure 8. The pan–tilt camera system was fixed to the ceiling, and rotated horizontally and vertically to construct the pan–tilt RGB-D scan dataset.

\subsection{5.2 Pan–Tilt RGB-D Scan Registration}

The proposed method was compared with four state-of-art methods that used different approaches. The first is the RGB-D camera calibration (RGBD-Calib) method [18], which makes use of external parameters of the camera for pose estimation. Both the proposed method and the RGB-D camera calibration method can be classified as calibration-based approaches, which require calibrations to be performed beforehand. For both methods, the same sequences of color and depth images were used to calibrate the rotation axes and external camera parameters. Two other methods are the FGR method proposed in [20] and the S4PCS method proposed in [19], which were compared with the RGB-D camera calibration method in [18]. Both methods are global registration methods, which can accept two point clouds of any status and globally search for the point correspondences to perform registration. The last method for comparison is the ORB-SLAM2 [2]. ORB-SLAM2 is one of the most cited SLAM methods, which can compute the trajectory of
monocular, stereo, and RGB-D cameras, as well as a sparse 3D reconstruction, provided that the stereo and RGB-D data with true scale is given. ORB-SLAM2 is also known to be a real-time SLAM library if CPU multi-threading is supported by the OpenMP library. However, as all other methods run in a single process/thread, multi-threading option for ORB-SLAM2 was disabled in the experiments for fairness.

The experiment and error metric were designed as were in [18]. The experiment process is as follows:

1) A pair of pan–tilt RGB-D scans is input to each algorithm, including a color image, a depth image, and point cloud data.
2) A rough transformation between two frames is estimated using the algorithm while its processing is measured.
3) Using the rough registration, the RMS error of $N$ closest points is measured.
4) The ICP algorithm is used to acquire the fine registration, with the rough estimation as the initial guess input, and the processing time is measured.
5) The RMS error of the fine registration is measured.

All algorithms except for the proposed were tested twice, first without ICP refinement and then with ICP refinement. The proposed algorithm was tested three times: (1) Only global registration, “Proposed (Default)”, (Section 4.2) without ICP refinement. (2) Global and local registration, "Proposed (Axis Bound)", (Section 4.3) without ICP refinement. (3) Global and local registration with ICP refinement. This is to better demonstrate the performance of Proposed (Default) with RGBD-Calib [18] in terms of global registration. Moreover, it is to demonstrate the improvement made in Proposed (Axis Bound) by local registration, over Proposed (Default), global registration. In total, eleven variations were derived from five algorithms with different options and tested in the experiments.

To quantitatively measure the performance of the registration results, the RMS metric of $N$ closest points was used [18]. Given $N$ points in the input point cloud, or frame $r$, their closest points are determined in the source point cloud, or frame $l$. If the vector storing the vector distances (in millimeters) between $N$ closest point pairs is denoted by $d_{min} \in \mathbb{R}^N$ and its $i$-th element by $d_{min}(i)$, the RMSE is defined as

$$RMS = \sqrt{\frac{\sum_{i=1}^{N} d_{min}^2(i)}{N}}. \quad (26)$$

All experiments were conducted on a desktop computer running Windows 10 with Intel Core i7-6700K CPU @ 4.00 GHz and 16 GB of DDR4 memory. For image processing and camera calibration, the OpenCV library [35] was used. For ICP algorithm and $N$ closest points RMSE implementation, the PCL library was used [36]. All codes are written in C++ with the same compiler optimization option and no multi-threading.

### 5.3 Experiment, Evaluation, and Analysis

Figure 9 shows eight pan–tilt scan datasets used for the RGB-D registration experiments. Each subfigure of Figure 9 is used for an independent experiment. In each subfigure, the point cloud on the left is input as frame $l$, and on the right point cloud as frame $r$. The eight test cases can be classified into three categories as in Table 1, i.e., pan, tilt, and pan–tilt, with respect to how the dataset was created.

The pan category includes subfigures (a)–(c) of Figure 9. The pan category was created by rotating the pan–tilt camera system in pan direction with increasing angles at each step, resulting in decreasing overlapping ratio between two scanned RGB-D frames. The tilt category includes subfigures (d)–(f) of Figure 9. The tilt category was created by rotating the pan–tilt camera system in the tilt direction with increasing angles at each step, resulting in decreasing overlapping ratio between two scanned RGB-D frames. Subfigures (g) and (h) of Figure 9 are classified as pan–tilt, which entails random rotations in both pan and tilt directions.

The registration results of input RGB-D scans (a)–(h) of
| Algorithm                      | Seconds | RMSE     |
|-------------------------------|---------|----------|
| Proposed (Axis Bound) w/ ICP  | 6.165796| 18.240154|
| RGBD-Calib w/ ICP             | 7.841424| 18.31772 |
| ORB-SLAM2 w/ ICP              | 11.518395| 18.290323 |
| FGR w/ ICP                    | 97.338219| 18.285046 |
| S4PCS w/ ICP                  | 507.913788| 33.473557 |

5.3.1 Algorithms with ICP refinement

The performance results of the algorithms with ICP refinement are first examined. In terms of accuracy, the proposed method with ICP, called “Proposed (Axis Bound) w/ ICP”, yielded the best RMSE results in five out of eight test experiments. In the experiments (d), (e), and (f), in which the proposed method did not rank first, “RGBD-Calib w/ ICP” yielded the best RMSE results. The datasets (d), (e), and (f) are categorized as “Tilt” in Table 1. The Tilt dataset was created with the camera facing directly at the front wall and tilting by certain amounts. As the checkerboard was placed in parallel to the wall in the camera calibration process in the RGBD-Calib method, it is conjectured that this planar-aligned calibration affected the external parameter calibration quality and, in turn, the registration results qualities. Nevertheless, the proposed method performed quite comparably even in such cases.

In general, it is quite apparent that the global registration methods, FGR and S4PCS, performed in general poorly even with ICP refinement. It becomes clearer if one examines figures in Appendix that the two global registration methods sometimes completely fail to perform reasonable registration. It is conceivable that the two algorithms are originally designed to match 3D scans of an object, thus performing poorly when registering two partial point clouds of a room geometry with low overlapping ratio and a large number of multiple occurring features such as corners and planes.

In terms of speed, in six out of eight experiments, the proposed method with ICP refinement required the least time to meet the termination requirement of the ICP algorithm. In the other two experiments, the proposed method with ICP refinement was the second fastest algorithm. It is noticeable that in several cases, the proposed method was twice as fast as the second and exhibited comparable or bet-
TABLE 3: Registration experiment results without ICP refinement. The shortest time to complete is highlighted in yellow. The lowest RMSE is highlighted in red. The algorithm with the lowest RMSE is emboldened. RGBD-Calib [18], ORB-SLAM2 [2], FGR [20], and S4PCS [19] are compared with the proposed method.

| Algorithm          | Scene 0 and 10 | Seconds | RMSE     | Algorithm          | Scene 0 and 20 | Seconds | RMSE     |
|--------------------|----------------|---------|----------|--------------------|----------------|---------|----------|
| Proposed (Default) | 0.000246       | 24.216434 |          | Proposed (Default) | 0.000152      | 39.480484 |          |
| Proposed (Axis Bound) | 0.216397     | 22.089104 |          | Proposed (Axis Bound) | 0.198129    | 36.073405 |          |
| RGBD-Calib         | 0.000253       | 28.540878 |          | RGBD-Calib         | 0.000229      | 29.886969 |          |
| ORB-SLAM2          | 1.074861       | 34.716242 |          | ORB-SLAM2          | 1.032496      | 33.239819 |          |
| FGR                | 97.661324      | 18.544088 |          | FGR                | 88.800407     | 43.18259  |          |
| S4PCS              | 506.601318     | 34.93003  |          | S4PCS              | 813.314758    | 26.734369 |          |

| Algorithm          | Scene 0 and 30 | Seconds | RMSE     | Algorithm          | Scene 0 and 45 | Seconds | RMSE     |
|--------------------|----------------|---------|----------|--------------------|----------------|---------|----------|
| Proposed (Default) | 0.000152       | 29.798969 |          | Proposed (Default) | 0.000153      | 22.617462 |          |
| Proposed (Axis Bound) | 0.182541     | 29.480431 |          | Proposed (Axis Bound) | 0.186479    | 32.014629 |          |
| RGBD-Calib         | 0.000229       | 28.540878 |          | RGBD-Calib         | 0.000229      | 28.540878 |          |
| ORB-SLAM2          | 0.997898       | 39.122582 |          | ORB-SLAM2          | 1.032496      | 33.239819 |          |
| FGR                | 89.779678      | 53.558205 |          | FGR                | 80.843987     | 18.279284 |          |
| S4PCS              | 470.615967     | 36.979393 |          | S4PCS              | 849.497375    | 36.19183  |          |

| Algorithm          | Scene 0 and 39 | Seconds | RMSE     | Algorithm          | Scene 0 and 49 | Seconds | RMSE     |
|--------------------|----------------|---------|----------|--------------------|----------------|---------|----------|
| Proposed (Default) | 0.00016        | 35.338062 |          | Proposed (Default) | 0.000157      | 40.294407 |          |
| Proposed (Axis Bound) | 0.187851     | 21.70833  |          | Proposed (Axis Bound) | 0.200695    | 39.18429  |          |
| RGBD-Calib         | 0.000268       | 28.540878 |          | RGBD-Calib         | 0.000267      | 33.239819 |          |
| ORB-SLAM2          | 0.989275       | 21.250082 |          | ORB-SLAM2          | 0.871645      | 29.009817 |          |
| FGR                | 84.577904      | 20.597624 |          | FGR                | 96.257271     | 52.473236 |          |
| S4PCS              | 470.615967     | 36.979393 |          | S4PCS              | 690.405457    | 32.846741 |          |

| Algorithm          | Scene 0 and 47 | Seconds | RMSE     | Algorithm          | Scene 0 and 50 | Seconds | RMSE     |
|--------------------|----------------|---------|----------|--------------------|----------------|---------|----------|
| Proposed (Default) | 0.000157       | 39.122582 |          | Proposed (Default) | 0.000155      | 37.162189 |          |
| Proposed (Axis Bound) | 0.193834    | 28.088261 |          | Proposed (Axis Bound) | 0.186479    | 32.014629 |          |
| RGBD-Calib         | 0.000245       | 38.540878 |          | RGBD-Calib         | 0.000245      | 32.599045 |          |
| ORB-SLAM2          | 1.060248       | 33.096642 |          | ORB-SLAM2          | 0.994338      | 43.82259  |          |
| FGR                | 99.504509      | 53.558205 |          | FGR                | 93.62558      | 52.473236 |          |
| S4PCS              | 1670.844849    | 31.75213  |          | S4PCS              | 220.25766     | 40.932125 |          |

5.3.2 Algorithms without ICP refinement
The algorithms without ICP refinement are now considered. The speed comparison between the two calibration-based approaches, "Proposed (Default)" and RGBD-Calib, is as follows. As explained above, Proposed (Default) derives the transformation estimation only from the servo control and its rotation transformation, i.e., Equations 13 and 8. By contrast, RGBD-Calib uses external parameters of the camera calibration to estimate camera poses. Even though the two algorithms have comparable RMSE performance, Proposed (Default) is always faster than RGBD-Calib. This is because the camera pose look-up table of RGBD-Calib requires $O(n)$ time (it was implemented with std::map::find of C++ STL), whereas Equations 13 and 8 of the proposed method require $O(1)$ time regardless of the number of pan–tilt camera poses used in the calibration.

The two feature-matching-based algorithms, Proposed (Axis Bound) and ORB-SLAM2, are also worth examining. Proposed (Axis Bound) has lower RMSE in seven experiments, except only for "(e) Scene 39 and 40". Moreover, Proposed (Axis Bound) completed in approximately 0.2 seconds, whereas ORB-SLAM2 in approximately 1 second. It is conceivable that the proposed axis bound registration method (Section 4.3.2) captures the geometric and kinetic characteristics of the pan–tilt rotations and exploits them in RGB-D scan registration, resulting in fast and accurate registration.

In conclusion, the proposed method achieved optimal results with and without ICP refinement. This implies that the proposed method can provide a valid starting point, or the "right track", that leads the ICP algorithm to better
results, preventing the algorithm from being stuck in a local optimum. In addition to providing accurate results, the proposed method was faster even with the ICP refinement process. This indicates that the proposed algorithm can produce an initial registration result that is very close to the optimal solution. Consequently, the proposed method performed at least comparably to, and often outperformed, other state-of-the-art registration algorithms in terms of computational accuracy (RMSE), speed, and thus efficiency.

6 Conclusion

Axis bound registration was proposed, which is a novel method for registering RGB-D scans of a pan–tilt–zoom camera. Through the rotation axis calibration and camera-servo control, the proposed method recovers the rotation parameters and its transformation model. The prior knowledge on the rotational motion of the pan–tilt–zoom servos leads to a novel distance-to-rotation-trajectory constraint that is used to reject false matching of keypoints between two frames without an iterative and time-consuming random sample consensus (RANSAC) process. The recovered transformation model is used to construct an objective function that minimizes the error in point-to-point matching. The (empirically one-step) alternating optimization scheme is adopted to divide the objective function into two independent problems, i.e., determining pan and tilt rotations, that can be solved in a closed form, and it ultimately accelerates optimization convergence. In the experiments, the proposed method was compared with four other state-of-the-art registration algorithms and tested for the RMSE of distances to N closest points and computation time. In five out of seven experiment conditions, the proposed method outperformed the other registration methods by great margins in terms of both RMSE and speed. This demonstrates that the proposed method can provide more accurate and efficient solutions for point cloud registration.

In the future, the proposed algorithm should be extended to incorporate online, dense, and volumetric reconstruction that continuously updates constructed 3D scenes with dynamic objects.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2018R1A2A1A05078628).

References

[1] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, “Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments,” in In the 12th International Symposium on Experimental Robotics (ISER). Citeseer, 2010.

[2] R. Mur-Artal and J. D. Tardö, “ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras,” IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255–1262, 2017.

[3] S. Ikehata, H. Yang, and Y. Furukawa, “Structured indoor modeling,” in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1323–1331.

[4] D. Kurz, F. Hantsch, M. Grobe, A. Schiewe, and O. Bimber, “Laser pointer tracking in projector-augmented architectural environments,” in Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on. IEEE, 2007, pp. 19–26.

[5] A. Wilson, H. Benko, S. Izadi, and O. Hilliges, “Steeerable augmented reality with the beamatron,” in Proceedings of the 25th annual ACM symposium on User interface software and technology. ACM, 2012, pp. 413–422.

[6] J.-H. Byun, S.-H. Chae, Y.-S. Yang, and T.-D. Han, “AIR: Anywhere Immersive Reality with User-Perspective Projection,” in EG 2017 - Short Papers. The Eurographics Association, 2017.

[7] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of rgb-d slam systems,” in Proc. of the International Conference on Intelligent Robot Systems (IROS), Oct. 2012.

[8] A. U. Nawoed, L. Chen, M. Dou, and H. Fuchs, “Enhancement of 3d capture of room-sized dynamic scenes with pan-tilt-zoom cameras,” in International Symposium on Visual Computing. Springer, 2014, pp. 379–389.

[9] R. Ambruș, N. Bore, J. Folkesson, and P. Jensfelt, “Meta-rooms: Building and maintaining long term spatial models in a dynamic world,” in Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on. IEEE, 2014, pp. 1854–1861.

[10] Z. Niu, K. Liu, Y. Wang, S. Huang, X. Deng, and Z. Zhang, “Calibration method for the relative orientation between the rotation axis and a camera using constrained global optimization,” Measurement Science and Technology, vol. 28, no. 5, p. 055001, 2017.

[11] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, “Kinectfusion: Real-time dense surface mapping and tracking,” in Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on. IEEE, 2011, pp. 127–136.

[12] P. J. Besl, N. D. McKay et al., “A method for registration of 3-d shapes,” IEEE Transactions on pattern analysis and machine intelligence, vol. 14, no. 2, pp. 239–256, 1992.

[13] D. Neumann, F. Lugauer, S. Bauer, J. Wasza, and J. Horneregger, “Real-time rgb-d mapping and 3-d modeling on the gpu using the random ball cover data structure,” in Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on. IEEE, 2011, pp. 1161–1167.

[14] J. Davis and X. Chen, “Calibrating pan-tilt cameras in wide-area surveillance networks,” in In IEEE International Conference on Computer Vision. Citeseer, 2003.

[15] Z. Wu and R. J. Radke, “Keeping a pan-tilt-zoom camera calibrated,” IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 8, pp. 1994–2007, 2013.

[16] P. Chen, M. Dai, K. Chen, and Z. Zhang, “Rotation axis calibration of a turntable using constrained global optimization,” Optik-International Journal for Light and Electron Optics, vol. 125, no. 17, pp. 4831–4836, 2014.

[17] J.-H. Byun, S.-H. Chae, and T.-D. Han, “Accurate Control of a Pan-tilt System Based on Parameterization of Rotational Motion,” in EG 2018 - Short Papers. The Eurographics Association, 2018.

[18] C.-Y. Tsai and C.-H. Huang, “Indoor scene point cloud registration algorithm based on rgb-d camera calibration,” Sensors, vol. 17, no. 8, p. 1874, 2017.

[19] N. Mellado, D. Aiger, and N. J. Mitra, “Super 4pcs fast global pointcloud registration via smart indexing,” in Computer Graphics Forum, vol. 33, no. 5. Wiley Online Library, 2014, pp. 205–215.

[20] Q.-Y. Zhou, J. Park, and V. Koltun, “Fast global registration,” in European Conference on Computer Vision. Springer, 2016, pp. 766–782.

[21] D. Aiger, N. J. Mitra, and D. Cohen-Or, “4-points congruent sets for robust pairwise surface registration,” in ACM Transactions on Graphics (TOG), vol. 27, no. 3. ACM, 2008, p. 85.

[22] A. J. Davison, “Real-time simultaneous localisation and mapping with a single camera,” in Computer Vision (ICCV), 2003 IEEE International Conference on. IEEE, 2003, p. 1403.

[23] G. Klein and D. Murray, “Parallel tracking and mapping for small ar workspaces,” in Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on. IEEE, 2007, pp. 225–234.

[24] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, “Dtam: Dense tracking and mapping in real-time,” in Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, 2011, pp. 2320–2327.

[25] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, “3-d mapping with an rgb-d camera,” IEEE Transactions on Robotics, vol. 30, no. 1, pp. 177–187, 2014.

[26] E. Mendes, P. Koch, and S. Lacroix, “Icp-based pose-graph slam,” in Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on. IEEE, 2016, pp. 195–200.
[27] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, “Elasticfusion: Real-time dense slam and light source estimation,” *Int'l. J. of Robotics Research*, IJRR, 2016.

[28] B. Schaffrin, “A note on constrained total least-squares estimation,” *Linear algebra and its applications*, vol. 417, no. 1, pp. 245-258, 2006.

[29] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE, 2011, pp. 2564–2571.

[30] S. Chae, Y. Yang, H. Choi, I.-J. Kim, J. Byun, J. Jo, and T.-D. Han, “Smart advisor: Real-time information provider with mobile augmented reality,” in *Consumer Electronics (ICCE), 2016 IEEE International Conference on*. IEEE, 2016, pp. 97–98.

[31] B. Bellekens, V. Spruyt, R. Berkvens, and M. Weyn, “A survey of rigid 3d pointcloud registration algorithms,” in *AMBIENT 2014: the Fourth International Conference on Ambient Computing, Applications, Services and Technologies, August 24-28, 2014, Rome, Italy*, 2014, pp. 8–13.

[32] S. R. Buss, “Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods,” *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[33] A. Kaehler and G. Bradski, *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. “ O'Reilly Media, Inc.”, 2016.

[34] Q.-Y. Zhou and V. Koltun, “Color map optimization for 3d reconstruction with consumer depth cameras,” *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 155, 2014.

[35] G. Bradski, “The OpenCV Library,” *Dr. Dobb's Journal of Software Tools*, 2000.

[36] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

Jung-Hyun Byun received the BS degree in computer science from Yonsei University, Seoul, Korea in 2015. Currently he is a Ph.D. candidate in the Department of Computer Science at Yonsei University, Seoul, Korea. His research interests lie in the fields of computer vision, computer graphics, machine learning and human-computer interaction. Particularly, his research focuses on realizing augmented reality environment, processing point cloud data and projection mapping.

Tack-Don Han received the PhD degree in computer engineering from the University of Massachusetts. He is a professor in the Department of Computer Science at Yonsei University, Korea. His research interests also include human-computer interaction and augmented reality.