Automatic markerless registration of point clouds with semantic-keypoint-based 4-points congruent sets

Xuming Ge

Chair of Geodesy, Technische Universität München, Arcisstraße 21, 80803 München, Germany

A R T I C L E   I N F O

Article history:
Received 20 October 2016
Received in revised form 5 April 2017
Accepted 29 June 2017
Available online 6 July 2017

Keywords:
Coarse registration
Terrestrial laser scanning technology
Point clouds
4PCS
K-4PCS

A B S T R A C T

The coarse registration of point clouds from urban building scenes has become a key topic in applications of terrestrial laser scanning technology. Sampling-based algorithms in the random sample consensus (RANSAC) model have emerged as mainstream solutions to address coarse registration problems. In this paper, we propose a novel combined solution to automatically align two markerless point clouds from building scenes. Firstly, the method segments non-ground points from ground points. Secondly, the proposed method detects feature points from each cross section and then obtains semantic keypoints by connecting feature points with specific rules. Finally, the detected semantic keypoints from two point clouds act as inputs to a modified 4PCS algorithm. Examples are presented and the results compared with those of K-4PCS to demonstrate the main contributions of the proposed method, which are the extension of the original 4PCS to handle heavy datasets and the use of semantic keypoints to improve K-4PCS in relation to registration accuracy and computational efficiency.

1. Introduction

Registration is an intermediate but crucial step in the three-dimensional (3D) reconstruction of real objects. A cloud of point samples from an object is typically obtained from two or more points of view in different reference frames. A prerequisite for further processing is thus to align all individual scans in a common coordinate system to obtain one large point cloud of the complete scene. This process of aligning all scans in a common reference system is called registration.

The registration strategy may differ depending on whether artificial targets (Ge and Wunderlich, 2015) are used to provide the reference points in two clouds of point sets. In terms of whether initial information is required, registration techniques can be classified as either coarse or fine registration. In coarse registration, the main goal is to compute an initial estimate of the rigid motion between two corresponding clouds; in fine registration, a higher-quality initial estimate is always required before the calculation. The scope of this paper will be limited to markerless coarse registration.

Markerless registration is achieved by using a sufficient overlap of the point clouds in different datasets and minimizing the sum of squares of the distance between the temporarily corresponding points in each iteration. For markerless fine registration, the iterative closest point (ICP) algorithm and its variations (Chen and Medioni, 1991, 1992; Besl and McKay, 1992; Zhang, 1994; Ge and Wunderlich, 2016a) are the commonly used standard approaches. Although the ICP method is a powerful algorithm for markerless registration, it has obvious shortcomings, e.g., it may converge to a local minimum or even be non-convergent without an priori alignment of the point clouds (Fusiello et al., 2002; Gruen and Acka, 2005; Salvi et al., 2007). Therefore, we use coarse registration to obtain a higher-quality initial estimate between scans, which should be sufficient for standard ICP to accomplish fine registration. In this paper, we are interested mainly in point clouds from urban building scenes. There are three key challenges in such coarse registration: (1) uneven point densities of the point clouds due to the polar measurement principle, (2) a vast amount of data (millions of points), and (3) it can involve a wide variety of objects (e.g., people and cars) with symmetric and incomplete structures. To address these challenges, many scientific studies have been carried out to register point clouds in building scenes (e.g., Brenner et al., 2008; Magnusson, 2009; Theiler and Schindler, 2012; Theiler et al., 2013, 2014a, 2015; Yang et al., 2016).

Here, we adapt the 4-points Congruent Sets (4PCS) approach for coarse registration, proposed by Aiger et al. (2008), to the aforementioned challenges. Fig. 1 shows the overall workflow of the
algorithm. First of all, we have to down-sample the raw scans and then separate ground and non-ground points. Secondly, the non-ground point cloud should be segmented into several layers in each of which we identify the interesting keypoints from the two-dimensional (2D) image. Next, we perpendicularly project these keypoints onto the ground point cloud so as to obtain semantic keypoints. Finally, we carry out a modified 4PCS algorithm with these semantic keypoints from both the source and target point clouds to find an alignment between the source and the target. Hereinafter, we will refer to the combined method as **Semantic-keypoint-based 4-Points Congruent Sets (SK-4PCS)**. We implement the SK-4PCS method in C++, making use of the open-source Point Cloud Library (PCL; Rusu and Cousins, 2011). The method of **Keypoint-based 4-Points Congruent Sets (K-4PCS)**, proposed by Theiler et al. (2014a), will be used as a reference method to demonstrate the properties of the SK-4PCS method. In Theiler et al. (2014a, 2014b), the authors show that the original 4PCS method is not able to register point clouds from large-scale scenes. Thus, the original 4PCS method is not used as a reference method in this paper.

2. Related work

Terrestrial laser scanners (TLS) are widely used to acquire dense 3D point clouds for various applications in geosciences, entertainment, cultural heritage, and urban planning. Hence, the topic of coarse registration is currently attracting great interest in the fields of computer vision and geodetic engineering. Numerous approaches have been published over recent years, which aim at the automatic registration of two or more point clouds. With ever-increasing scanning resolutions, it is necessary to represent raw point clouds before registration. Thus, coarse registration algorithms consist of two main steps: (1) representation, i.e., raw point clouds are reduced to sparse sets of features; (2) matching and transformation, i.e., correspondences are established between two point clouds in overlapping areas and are then used to estimate the transformation parameters that align the point clouds sufficiently well.

Automatic coarse registration methods can be classified into four categories—point-based, line-based, plane-based, and structure-based—in terms of the type of features used for registration. In general, the point-based method is the most popular strategy in coarse registration. Böhm and Becker (2007) investigated automatic markerless registration with TLS data by using SIFT keypoints (Lowe, 2004). Kang et al. (2009) carried out a similar scheme to complete automatic registration and added a strategy to detect and remove outliers between feature correspondences. Another popular line of thought is to derive features by calculating a point’s characteristic from its neighborhood, e.g., the Point Feature Histograms (PFH; Rusu et al., 2008) method that estimates a set of robust 16D features for a keypoint from its K-nearest neighborhood, and its accelerated version (FPFH; Rusu et al., 2009). Keypoint detectors are also common tools for detecting features in 3D data. Theiler et al. (2014a) proposed a markerless registration scheme that matches the extracted difference of Gaussians (DoG) or Harris keypoints. The authors also provided an updated version by using DoG and Harris keypoints in registration (Theiler et al., 2014b). Semantic feature points represent another type of keypoint. Yang et al. (2016) proposed an automatic coarse registration method by using semantic feature points. Compared with the point-based method, the other strategies (i.e., line-based, plane-based, and structure-based methods) are limited more to man-made environments in relation to extraction features (e.g., Dold and Brenner, 2006; Von Hansen, 2006; Brenner et al., 2008; Theiler and Schindler, 2012; Yang and Zang, 2014).

![Fig. 1. Point-cloud registration framework.](image)
Aiger et al. (2008) proposed an $O(n^2 + k)$ RANSAC algorithm that uses a 4-point base (i.e., 4PCS), where $k$ is the number of congruent bases. Mellado et al. (2014) proposed a method called Super 4PCS to reduce the runtime complexity to $O(n + k_1 + k_2)$, where $k_1$ is the number of pairs of a given distance and $k_2$ is the number of congruent bases. Mohamad et al. (2014) introduced a generalized 4PCS (G-4PCS) algorithm in which the authors exploited the richer geometry of 3D data and generalized the 4-point base by removing the planarity constraint. An accelerated version of G-4PCS (i.e., Super G-4PCS) was presented by Mohamad et al. (2015). Theiler et al. (2014a) presented a combined 4PCS method, called Keypoint-based 4PCS (K-4PCS), in which keypoints are used to represent raw point clouds. Moreover, Theiler et al. (2014b) presented an updated version of the K-4PCS algorithm. Since the number of points is significantly reduced before matching and transformation in K-4PCS, the calculation is significantly quicker when compared with other versions of the 4PCS algorithm. Ge (2016b) presented a GD-4PCS (i.e., Geodesic distances 4PCS) method to handle point clouds matching that extended the 4PCS method from rigid cases to non-rigid cases.

3. Point clouds representation

3.1. Down-sampling and segmentation

With ever-increasing scanning resolutions, it is beneficial to down-sample huge raw point clouds, potentially consisting of millions of points, before coarse registration. Therefore, a voxel grid filter is first applied in our approach. The whole scan volume is divided into a regular 3D voxel grid, and only one point per voxel of size $\tau_v$, computed as the centroid of the points inside a grid cell, is retained. The filtered point cloud serves as input to the semantic keypoint detectors. After obtaining down-sampled point clouds, we need to separate the ground points and non-ground points both in the source and target before keypoint detection. Various approaches can be applied here to segment ground points, e.g., Zhang et al. (2003) and Hernández and Marcotegui (2009). In our tests, we exploited the approach of Zhang et al. (2003) to remove ground points (see Fig. 2).

3.2. Feature-point extraction in 2D images

Typically, salient geometric entities are used to generate feature points. There are three main schemes to extract feature points from such entities: (1) randomly select a small set of features, e.g., Leng et al. (2014); (2) use 3D keypoint detectors, e.g., Theiler et al. (2014); (3) exploit man-made structures, e.g., Yang et al. (2016). We applied a similar strategy to that of Yang et al. (2016) to extract feature points from structure information. The difference here is that we projected each cross-section point cloud onto a specific plane and then extracted feature points from different 2D images by using a bespoke RANSAC framework.

The non-ground points are segmented into cross sections by a series of horizontal planes with height interval $\delta_h$ (see Fig. 3, in which the cross section is dotted in green). Fig. 4 shows the inside of the structure; ground points, non-ground points, and cross sections are dotted in blue, red, and green, respectively. The thickness $\delta_t$ of each cross section is predefined according to $\tau_v$ in the voxel filter. Usually, we can define $\delta_t = 2\tau_v$ to avoid holes in a cross section. Typically, the contour of a geometric entity can be reflected by lines (including straight lines and curves) in a series of continuous cross sections if $\delta_t$ and $\delta_h$ are both set to reasonable values. For each cross section, we projected those points onto a corresponding specific horizontal plane with Z-dimension value equal to the mean value of those points in the Z axis, and then treated them as a 2D image. In 2D images, straight lines and circles can be easily detected and segmented. Thus, we run the bespoke RANSAC framework to obtain such structures from each cross section (see Fig. 5). In such a framework, some thresholds should be predefined so as to control the quality of the results, i.e., the fitting threshold $\delta_f$ of a line, the minimal cluster range $l_c$ between two lines, the fitting threshold $\delta_c$ of a circle, and the maximum radius $r_{\text{max}}$ of the fitted circle. After obtaining all straight lines and circles in a cross section, we can continue to obtain the vertexes of lines, vertexes of arcs, intersection points of two lines, and centers of circles. Fig. 5 shows the vertexes of lines (red points) and intersection points of two lines (green points) from a cross section.

3.3. Semantic keypoint extraction

After obtaining the feature points in all cross sections, we need to distinguish and connect them. First of all, we need to distinguish feature points in each cross section by using different semantic information. For each feature point, we mark different labels according to their location and potential location, e.g., beginning or ending of vertexes of lines, intersection points of two lines, and centers of circles (see Fig. 5). Technically, the more classifications we have, the more advantages for later transformation estimation. Since the reference heights in two point clouds may be different, the cross sections may not have one-to-one correspondence. Thus, a public reference plane is needed, and in our program the ground acts as such a plane. All the feature points are connected vertically by different vertical lines that are all perpendicu-

![Fig. 2. Segmentation of ground points (green) and non-ground points (red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)](image-url)
lar to the XY-plane. A threshold $\rho_{\text{max}}$ should be given here to restrict the maximum distance from a feature point to a vertical line. Moreover, we further regulate that, for each vertical line, only one feature point can contribute to it in one cross section. Next, for each vertical line, we can obtain an intersection point between the line and the ground plane (see Fig. 6). Subsequently, all the semantic information (i.e., the number of feature points and the label of each feature point) from that vertical line can be stored in the intersection point. Finally, we obtain a series of intersection points that are marked as semantic keypoints in this paper.

---

**Fig. 3.** Cross-section segment (green points) from the whole point cloud. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

**Fig. 4.** Inside view of the segmentation. Cross section (green), non-ground points (red), and ground points (blue). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

**Fig. 5.** Extraction of feature points from a cross section. Vertexes of lines (red points) and intersection points of two lines (green points). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
4. Semantic-keypoint-based 4PCS matching

The 4PCS algorithm (Aiger et al., 2008) is an efficient global rigid registration algorithm for 3D point sets. It was designed for aligning partially overlapping but rather evenly distributed, sparse point clouds with arbitrary orientations. The main goal of 4PCS is to find the transformation that provides the best alignment as measured by the greatest amount of overlap between the source point cloud and the target point cloud. The algorithm is based on finding a set of 4-point bases in the source cloud that are congruent to a 4-point base selected from the target cloud. Letting $T = \{a, b, c, d\}$ be four coplanar points selected from the target cloud, if those four points are not all collinear then the line $ab$ intersects the line $cd$ at an intermediate point $e$; see Fig. 7. Two ratios can then be defined when we have a 4-point base constructed from two intersection pairs:

$$r_1 = \frac{|c-e|}{|a-b|}$$
$$r_2 = \frac{|a-c|}{|e-d|}$$

(1)

There ratios are invariant under affine transformation (Huttenlocher, 1991) and therefore act as invariants constraining the search for congruent 4-point bases in the source cloud. In terms of rigid registration, as suggested by Aiger et al. (2008), one can add additional constraints to accelerate the searching process, i.e.,

$$\|a_1 - b_1\| - \|a - b\| < e_1$$
$$\|c_1 - d_1\| - \|c - d\| < e_1$$

(2)

where $e_1$ is a given distance tolerance and $T_1 = \{a_1, b_1, c_1, d_1\}$ is a corresponding base from the source cloud. As did Theiler et al. (2014a), we can further strengthen this constrain, i.e.,

$$\|a_1 - c_1\| - \|a - c\| < e_2$$
$$\|c_1 - b_1\| - \|c - b\| < e_2$$
$$\|a_1 - d_1\| - \|a - d\| < e_2$$
$$\|d_1 - b_1\| - \|d - b\| < e_2$$

(3)

with a given distance tolerance $e_2$. Connecting Eqs. (2) and (3), we can unify the distance tolerance level by letting $e_1 = e_2$. Mellado et al. (2014) accelerated the searching process by predefining the distance $d = |a - b| = |c - d|$.

Mohamad et al. (2014) extended the 4PCS approach to real 3D searching in order to find 3D intersection points. In G-4PCS, it is no longer necessary to restrict the 4-point bases to those from 3D models that are necessarily planar. Fig. 8 shows the extraction of 3D intersection points from a non-coplanar 4-point base $S = \{p, q, i, j\}$. In the 4-point base $S$, the line $pq$ intersects the line $ij$. The line $mp$ intersects the line $in$.

![Fig. 6. Samples of semantic keypoint extraction from a panoramic photograph of the scanning field. Blue lines: vertical lines that connect feature points; red points: vertexes of lines; green points: intersection points of two lines; yellow points: potential intersection points; pink triangles: semantic keypoints. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)](image1)

![Fig. 7. Illustration of the intersection point $e$ in the selected base $T$.](image2)

![Fig. 8. Illustration of the 3D intersection points $m$ and $n$ in the selected base $S$.](image3)
in 3D at a point $m$ on $pq$ and $n$ on $ij$, such that $mn$ is the shortest distance between $pq$ and $ij$. Equation (1) can then be represented as

$$r_1 = \frac{|m - p|}{|p - q|},$$

$$r_2 = \frac{|n - p|}{|p - j|}.$$  

Considering the property in rigid registration, Eqs. (2) and (3) can also be exploited. Moreover, an additional constrain can be added, i.e.,

$$d_3 = |m - n|.$$  

In the 4PCS algorithm, symmetry problems arise (Aiger et al., 2008) if the overlapping areas exist in symmetrical parts. Since the selected four points are forced to be coplanar and intersecting in 4PCS, the symmetry problems are not serious unless there are many symmetrical structures in the scanned object. However, the symmetry problems are serious for 3D intersection points; see Fig. 9a. If the matched base from the source cloud is $T = (i, j, p, q)$ with the 3D intersection points $(n, m)$, then we can find many symmetrical bases that all satisfy Eqs. (2)-(5), e.g., $T_1 = (i, j, p_1, q_1)$, $T_2 = (i, j, p_2, q_2)$, and $T_3 = (i, j, p_3, q_3)$ with the 3D intersection points $(n_1, m)$, $(n_2, m)$, and $(n_3, m)$, respectively. Thus, it is necessary to remove the ambiguity in the base $T$. We divide the symmetry problems into vertical-level and horizontal-level ones, and handle them separately. For the vertical-level symmetry problems (e.g., $T_1$ with $T_2$), we calculate normal vectors $nm$ and $nm_1$, and then use the direction of the ground’s normal vector to separate and mark them. For the horizontal-level symmetry problems (e.g., $T_2$ with $T_3$), we firstly project $m_1$ and $m_2$ onto the horizontal plane with $n$, $i$, and $j$; see Fig. 9b. Next, we define a reference direction by the normal vector $ij$

and calculate azimuth angles $\alpha_2$ and $\alpha_3$, subsequently. Thus far, we can uniquely determine a matched 4-points base.

In the presented SK-4PCS algorithm, only the semantic keypoints serve as inputs to the correspondence searching process. This is different from the feature points that were detected by using 3D detectors (Theiler et al., 2014a, 2014b) because each semantic keypoint contains a specific meaning. Therefore, as well as satisfying Eqs. (2)-(5), the correspondences in SK-4PCS should also have the same semantic meanings. In other words, the matched bases $T$ and $S$ that are captured from the target cloud and source cloud, respectively, have to fulfill the geometric constraints and also satisfy the semantic requirements. Since each point has a specific meaning in SK-4PCS, the program that searches for matched bases in the source cloud does not have to use a totally brute-force searching process (Aiger et al., 2008; Theiler et al., 2014a). Nevertheless, in the SK-4PCS approach, 4-point bases are selected randomly as in the 4PCS, G-4PCS, and K-4PCS approaches, but we store each used base with its semantic information. The loop for seeking candidate-matched bases in SK-4PCS is described in Algorithm 1. A score value (Aiger et al., 2008), namely the fraction of points in the source cloud for which a match is found in the target cloud, can be calculated for each set of estimated transformation parameters. The iteration is terminated if a score value is lower than a given threshold.

5. Experimental evaluation

We experimentally evaluate the previously described SK-4PCS registration method with respect to geometric registration accuracy and computational efficiency. The K-4PCS method (Theiler et al., 2014a), which is an effective method for solving coarse registration, is used to provide the reference results to demonstrate...
the performance of the presented SK-4PCS method. The K-4PCS program is supported by the open-source PCL. Moreover, the K-4PCS algorithm in our experiments is the updated version proposed by Theiler et al. (2014b) but without the multi-threading strategy in order to speed up computation. The implementation of Theiler et al. (2014b) uses the OpenMP application programming interface to distribute separate trials of K-4PCS to different threads in order to significantly improve the computational efficiency. However, the improvement in computational efficiency by using the multi-threading strategy of the presented SK-4PCS method is the same as that found by Theiler et al. (2014b). The geometric registration accuracy is measured with the root-mean-square error (RMSE) between the true correspondences of the source point cloud \( Q \) and the transformed target point cloud \( P \) after coarse registration. The true correspondences are captured from artificial targets that we installed in the scanning scene. For a single-time transformation, we calculate the RMSE value as

\[
RMSE_i = \sqrt{\frac{1}{n} \sum_{k=1}^{n} (q_k - T(p_k))^2}
\]

where \( n \) is the number of true correspondences, and \( q_k \) and \( p_k \) are the \( k \)th corresponding points from the source and target clouds, respectively. Operator \( T(\cdot) \) is the rigid-body transformation operator and \( f(\cdot) \) is the point error operator. In general, the selected 4-point base and matched bases are captured randomly from two point clouds such that the 4PCS registration results are not constant. In other words, there are many sufficient correspondences of 4-point bases between the source and target clouds, and those different correspondences of 4-point bases lead to different registration results in terms of accuracy. So far, to evaluate such a 4PCS method, we need to run the program \( N \) times \((N > 1)\) and use the mean value to reflect the performance, i.e.,

\[
R = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (RMSE_i)^2}
\]

The computational efficiency is represented by the mean value \( R \) of \( N \) CPU runtimes. In order to interpret the comparisons between SK-4PCS and K-4PCS. All experiments are carried out on a standard 64-bit Windows 7 desktop computer with 8 GB RAM and an Intel (R) Core (TM) i5-4570S 2.90 GHz CPU.

5.1. Data description

We tested SK-4PCS on four different TLS datasets. The first three datasets were captured using a Leica HDS7000 laser scanner system with a field of view of 360° in the horizontal direction and 320° in the vertical direction, a maximum range of approximately 180 m, and a data rate of 127,000 points/s (i.e., a high-resolution, high-quality scan model). The last dataset was obtained using a Faro 3D x330 laser scanner system with a field of view of 360° in the horizontal direction and 300° in the vertical direction, a maximum range of approximately 330 m, and a data rate 976,000 points/s at 307 m.

The first dataset consists of two scans acquired in a square lobby with the approximate dimensions of 12 m \( \times \) 12 m \( \times \) 20 m (see Figs. 3 and 6) from different points of view. For each view, the scanner captured more than 25 million points. In this dataset, there are many fixed objects with regular structures that can make it easier for the presented method to extract semantic keypoints. Moreover, the scanning scenes are relatively open so the areas overlap by as much as 90%. Registration might be considered to be easier with such a high rate of overlapping. However, the main challenge in this case is that the architecture is strongly symmetrical, which might cause the matching process to fail. The second dataset was captured from a laboratory with the approximate dimensions of 20 m \( \times \) 10 m \( \times \) 5 m (see Fig. 2); more than 30 million points were captured in each scan. This laboratory contains objects such as office furniture (e.g., tables, chairs, shelves, and boards), some instruments and tools, and several pillars. We carried out two scans from different locations, but some objects were moved between the scans such that the overlap was reduced to 60–65%. Thus, the main challenge, in this case, is the low overlapping rate. Moreover, the point–cloud densities are uneven because of the asymmetry in the horizontal dimensions (i.e., 20 m \( \times \) 10 m). The third dataset was captured half a year later than the second one but from the same laboratory. The laboratory was being renovated when we scanned. So, there were wirings and pipes. Some tables and chairs were moved between the scans. Moreover, there were lots of odds and ends. We also carried out two scans from different locations. Because of the limitation of conditions, a part of the laboratory was obstructed from the viewpoint of the second scan. Thus, the overlapping rate was even lower than the second case. The last dataset consists of three scans acquired in a part of

![Symmetry problem in 3D intersection points](image)
building that is under construction with the approximate dimensions of 50 m × 30 m × 4 m; more than 80 million points were captured in each scan. One of the main challenges, in this case, is the huge dataset. Secondly, the overlapping rate between connected scans is about 50–60%, and a large proportion of those areas are the ground.

5.2. Generation parameters

As described in Sections 3 and 4, SK-4PCS has certain global parameters that have to be set by the user before registration. Setting the global parameters before registration to control the qualities of the extracted features is one of the most effective strategies in coarse registration (e.g., Brenner et al., 2008; Magnusson, 2009; Theiler et al., 2014a, 2014b; Yang et al., 2016). The global-parameter settings depend mainly on the point-cloud density, the size and complexity of the scene, and the computational cost. Furthermore, in most cases the global-parameter settings will significantly influence the accuracy of a coarse registration. In order to improve the operability and repeatability of SK-4PCS, we related the global parameters for semantic keypoint extraction to the size \( r_v \). We used the same strategy as that of Aiger et al. (2008) and Theiler et al. (2014a,b) to set the global parameters in the matching and transformation processes. Thus, in Table 1 we list only the global parameters for semantic keypoint extraction in our tests.

5.3. Lobby data set

Fig. 6 shows a part of the scanning scene in a panoramic photograph. Fig. 10 shows the point clouds of this architecture that were captured from two different locations, and Fig. 11 displays the registered point clouds using SK-4PCS.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Parameter settings for experimental datasets.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedure</td>
<td>Parameter</td>
</tr>
<tr>
<td>Semantic feature point extraction</td>
<td>( r_v )</td>
</tr>
<tr>
<td></td>
<td>( \delta_i )</td>
</tr>
<tr>
<td></td>
<td>( \delta_t )</td>
</tr>
<tr>
<td></td>
<td>( l_d )</td>
</tr>
<tr>
<td></td>
<td>( l_c )</td>
</tr>
<tr>
<td></td>
<td>( r_d )</td>
</tr>
<tr>
<td></td>
<td>( r_{max} )</td>
</tr>
<tr>
<td></td>
<td>( s_{max} )</td>
</tr>
<tr>
<td></td>
<td>( l_{min} )</td>
</tr>
</tbody>
</table>

Fig. 10. Raw point clouds of the lobby that were captured from two different locations.

Fig. 11. Registered point clouds of the lobby using SK-4PCS.
We set \( v = 8 \) cm to execute 50 times by separately using the SK-4PCS and K-4PCS methods (using a Harris detector with a window size of 3 \( v \)). Fig. 12 shows the CPU time for each iteration by using two different methods. Overall, SK-4PCS and K-4PCS have similar computational efficiencies. The mean CPU time \( \bar{t} \) in K-4PCS was 7.5 s and that in SK-4PCS was 6.7 s, an improvement of 11%. Fig. 13 shows the details of CPU runtimes for each SK-4PCS strategy. From Fig. 13, we see that the most consuming part was the down-sampling program, which required approximately half the time because of the huge inputs. The runtimes in the ground

Fig. 12. CPU time in each iteration using K-4PCS (blue) and SK-4PCS (pink). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 13. CPU runtimes for each K-4PCS strategy.

Fig. 14. Changes in mean CPU time with different sizes of voxel grid.
segmentation and the semantic keypoint extraction were relatively stable. The deviation of runtime in SK-4PCS is mainly from the 4PCS transformation calculation program. The deviation of runtime in SK-4PCS was 2.0 s and that in K-4PCS was 1.9 s. From Fig. 13, we find that if we assume that tests 9, 25, 35, and 46 are outliers (i.e., CPU time > 8 s) and remove them from the results, the deviation of runtime in SK-4PCS reduces to approximately 1 s. That means that SK-4PCS has a stable computational efficiency with a success rate of as much as 92%. If we reduce the size of the voxel grid from 8 cm to 2 cm, the CPU runtimes in SK-4PCS increase by 85% from approximately 7–13 s, but those in K-4PCS increase by 350% from 8 s to 36 s; see Fig. 14. This situation can be explained by the fact that the number of semantic keypoints was relatively stable but the number of 3D keypoints in K-4PCS increased significantly as the size of the voxel grid was decreased. We installed 15 black-and-white targets in the overlapping areas of the two scanning scenes, and used the accompanying software, Leica Cyclone, to extract the target centers from the point clouds. Based on Eqs. (6) and (7), we can obtain the mean value of RMSE, $\bar{R}$, for each method. The registered information is summarized in Table 2.

### Table 2

<table>
<thead>
<tr>
<th>$\tau_v$ = 8 cm</th>
<th>K-4PCS</th>
<th>SK-4PCS</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau$ (sec)</td>
<td>7.5 ± 1.9</td>
<td>6.7 ± 2.0</td>
<td>11%</td>
</tr>
<tr>
<td>$\bar{R}$ (cm)</td>
<td>1.2 ± 0.3</td>
<td>0.9 ± 0.2</td>
<td>25%</td>
</tr>
</tbody>
</table>

### 5.4. Laboratory data set I

Fig. 2 shows one of the laboratory point clouds after segmentation of the ground points. Fig. 15 shows the point clouds of the laboratory that were captured from two different locations, and Fig. 16 displays the registered point clouds by using SK-4PCS.

We set $\tau_v = 2$ cm to carry out the coarse registration by using SK-4PCS and K-4PCS (using a Harris detector with a window size of 3 $\tau_v$). Since the environment is more complex than in the lobby scenario, we reduced the size of the voxel grid in order to preserve features as completely as possible. Fig. 17 shows the CPU time in each iteration by using SK-4PCS and K-4PCS. The mean CPU time $\bar{t}$ in K-4PCS was 114.8 s and that in SK-4PCS was 43.1 s, an improvement of 62%. Fig. 18 further displays the specific time consumption of SK-4PCS in 100 registrations. Comparing Figs. 18 and 13, we find that the time consumptions significantly increased in semantic keypoint extraction as a result of the scene complexity. The deviation in runtime comes mainly from the 4PCS transformation calculation, it being as much as 15.7 s in SK-4PCS. The key factor that leads to this high deviation is the existence of low-overlapping areas. As we mentioned before, the overlapping rate is 60–65%, but this value may decrease if we consider only the semantic keypoints in the calculation. The low overlapping rate also infects the performances of K-4PCS, i.e., the deviation in runtime was 14.7 s. In Fig. 18, if we remove the iterations with runtimes of greater than 60 s (i.e., treat these as outliers), the value of $\bar{t}$ in SK-4PCS drops to 38.6 s and the deviation drops to 7.8 s, i.e., improvements of 10% and 50%, respectively. In this case, SK-4PCS has a relatively
stable computational efficiency with a success rate of as much as 90%. In the experiment, we installed 20 black-and-white targets in the overlapping areas of the two scanning scenes and used the accompanying software, Leica Cyclone, to extract the target centers from the point clouds. Based on Eq. (6), we calculate the RMSE value for each registration; see Fig. 19. The mean values of RMSE, $\bar{R}$, are 5.73 cm and 1.51 cm by using K-4PCS and SK-4PCS, respectively, in 100 registrations. The deviation in RMSE is 2.94 cm in K-4PCS but only 0.49 cm in SK-4PCS. The registered information is summarized in Table 3. It is undoubtedly the case that the large gap in computational efficiency here between K-4PCS and SK-4PCS is caused by the small size of the voxel grid. If we enlarge the voxel grid, the number of 3D keypoints is reduced and the computational efficiency is improved simultaneously. However, we should point out that increasing the size of the voxel grid leads to some features being lost, which may result in even lower registration accuracy.

<table>
<thead>
<tr>
<th>No. of tests</th>
<th>CPU time [']</th>
<th>0</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>114.8 ± 14.7</td>
<td>43.1 ± 15.7</td>
<td>62%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>15.3 ± 3.6</td>
<td>4.3 ± 1.4</td>
<td>74%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>10.5 ± 2.4</td>
<td>1.5 ± 0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3

<table>
<thead>
<tr>
<th>Results for the laboratory dataset I.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_c$ = 2 cm</td>
</tr>
<tr>
<td>(sec)</td>
</tr>
<tr>
<td>$\bar{R}$ (cm)</td>
</tr>
</tbody>
</table>

Fig. 17. CPU time in each iteration by using K-4PCS (blue) and SK-4PCS (pink). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 18. Specific time consumption of SK-4PCS in 100-times registration.

Fig. 19. RMSE performance in 100-times registration by using K-4PCS (blue) and SK-4PCS (red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
5.5. Laboratory data set II

Fig. 20 shows the registered point clouds by using SK-4PCS. From Fig. 20 we can clearly see that the pink point cloud (from the 2nd location) was shorter than the green one (from the 1st location) as we discussed before. The pipes mainly appeared in the second scan (see Fig. 20). We also set $\tau_p = 2$ cm as the second case to carry out 100 times of the coarse registration by using SK-4PCS and K-4PCS (using a Harris detector with a window size of 3 $\tau_p$) respectively. Totally, the case 3 and case 2 have similar computational efficiency. The mean CPU time $\overline{t}$ in K-4PCS was 104.3 s and that in SK-4PCS was 41.8 s, an improvement of 60%. Here, we did not further show the specific time consumption of SK-4PCS because it has a similar performance as that in case 2. In this case, SK-4PCS has a relatively lower computational efficiency, with a success rate of as much as 88%, when compared with the case 2.

In the experiment, we extracted 16 black-and-white targets in the 2nd location, 15 in the 1st location, and the green one was obtained from the second location, and the green one was scanned from the third location. Since this paper only considered targets. Fig. 22 shows the registered point clouds in different scans accompanying software, SCENE LT, we can obtain the transformed point clouds i.e. $T_{2i}$ and $T_{3i}$ (see Figs. 21 and 22). We can also gain the transformed point clouds $T_{2i}^{sk}$ and $T_{3i}^{sk}$ after i-th time R1 and R2 respectively by using SK-4PCS. In this experiment, we used an alternative strategy to calculate the RMSE values because the number of targets was limited. $T_2$ and $T_3$ were treated as the corrected registrations results then for each point in $T_{2i}^{sk}$ we can find its corresponding nearest point in $T_2$ and marked its distance as $\sqrt{2 - \| \}$.

We calculate the RMSE value in i-th time R1 by using SK-4PCS as

$$\text{RMSE}_m = \sqrt{\frac{1}{n} \sum_{m=1}^{n} (\sqrt{2 - \|}^2)}$$

where $m$ represents the number of points. After that, we can future to gain $R$ for 100 R1 based on Eq. (7). The same strategy can be applied to calculate $R$ in K-4PCS. The mean values of RMSE, $\overline{R}$, are 6.43 cm and 2.32 cm by using K-4PCS and SK-4PCS, respectively, in 100 registrations. The deviation in RMSE is 2.55 cm in K-4PCS but only 0.67 cm in SK-4PCS. The registered information is summarized in Table 4.

Table 4

<table>
<thead>
<tr>
<th>$\tau_p$ (cm)</th>
<th>K-4PCS</th>
<th>SK-4PCS</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\overline{t}$ (sec)</td>
<td>104.3 ± 9.5</td>
<td>41.8 ± 9.3</td>
<td>60%</td>
</tr>
<tr>
<td>$\overline{R}$ (cm)</td>
<td>6.43 ± 2.55</td>
<td>2.32 ± 0.67</td>
<td>64%</td>
</tr>
</tbody>
</table>

5.6. One floor of building data set

Fig. 21 displays the registered point clouds by using the accompanying software of Faro, Scene LT, based on the artificial sphere targets. Fig. 22 shows the registered point clouds in different scans i.e. the yellow point cloud was captured from the first location, the pink one was obtained from the second location, and the green one was scanned from the third location. Since this paper only considered the pairwise registration, in this experiment we separately carried out two independent registrations i.e. firstly transformed from location 2 (L2) to location 1 (L1) and the next from location 3 (L3) to L1. We marked these two registrations as R1 and R2. Fig. 23 shows the point clouds before and after R1 by using SK-4PCS. Here we did not further show the point clouds in R2. In this case, we also set $\tau_p = 2$ cm to execute 100 times by separately using the SK-4PCS and K-4PCS methods. The mean CPU runtimes, both in R1 and R2, in SK-4PCS were significantly less than that in the cases 2 and 3. The main factor is that the structures in the cross sections are simple. Moreover, for the same reason, the number of semantic keypoints in SK-4PCS was significantly less than the number of keypoints in K-4PCS. Thus, SK-4PCS also performed better than K-4PCS in terms of the CPU runtimes (see Table 5). We installed 4 sphere targets in the overlapping areas of connected scanning scenes such that to register point clouds. Based on the accompanying software, SCENE LT, we can obtain the transformed point clouds i.e. $T_{2i}$ and $T_{3i}$ (see Figs. 21 and 22). We can also gain the transformed point clouds $T_{2i}^{sk}$ and $T_{3i}^{sk}$ after i-th time R1 and R2 respectively by using SK-4PCS. In this experiment, we used an alternative strategy to calculate the RMSE values because the number of targets was limited. $T_2$ and $T_3$ were treated as the corrected registrations results then for each point in $T_{2i}^{sk}$ we can find its corresponding nearest point in $T_2$ and marked its distance as $\sqrt{2 - \|}$. We calculate the RMSE value in i-th time R1 by using SK-4PCS as

$$\text{RMSE}_m = \sqrt{\frac{1}{n} \sum_{m=1}^{n} (\sqrt{2 - \|}^2)}$$

where $m$ represents the number of points. After that, we can future to gain $R$ for 100 R1 based on Eq. (7). The same strategy can be applied to calculate $R$ in K-4PCS. The mean values of RMSE, $\overline{R}$, are 4.24 cm and 3.84 cm by using K-4PCS and SK-4PCS, respectively, in 100 R1. The deviation in RMSE is 1.2 cm in K-4PCS but only 0.3 cm in SK-4PCS. The registered information is summarized in Table 5.

5.7. Discussion and limitations of SK-4PCS

From the aforementioned, it can be seen that in the given experiments the proposed method, SK-4PCS, performs better than K-4PCS in terms of the registration accuracy. One reason for this is that in SK-4PCS we use semantic information to express each key point such that to increase the accuracy of the correspondences that will directly influence the registration accuracy. In other words, SK-4PCS makes the correspondences candidates from the regular to specific when compared with K-4PCS. This improvement is similar when we compared K-4PCS with the original 4PCS i.e. the candidates from the random to regular. Another reasonable interpretation is that as already mentioned a reference plane (e.g. the ground) is required in SK-4PCS that means a restriction will be used in the SK-4PCS computation. It is well known that accurate restrictions can increase the accuracy of adjustments.
Fig. 21. Registered point clouds of the one floor of building dataset by using sphere targets.

Fig. 22. Registered point clouds from 3 scans. The yellow, pink and green point clouds were scans from the first, second and the third locations respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 23. Registered point clouds by using SK-4PCS.

Table 5
Results for the building dataset.

<table>
<thead>
<tr>
<th>Locations</th>
<th>$\tau_p = 2$ cm</th>
<th>K-4PCS</th>
<th>SK-4PCS</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1–L2</td>
<td>/ (sec)</td>
<td>13.7 ± 2.5</td>
<td>7.3 ± 1.1</td>
<td>47%</td>
</tr>
<tr>
<td></td>
<td>$R$ (cm)</td>
<td>4.24 ± 1.2</td>
<td>3.84 ± 0.3</td>
<td>9%</td>
</tr>
<tr>
<td>L1–L3</td>
<td>/ (sec)</td>
<td>11.3 ± 2.1</td>
<td>7.0 ± 1.3</td>
<td>38%</td>
</tr>
<tr>
<td></td>
<td>$R$ (cm)</td>
<td>4.04 ± 1.0</td>
<td>3.14 ± 0.3</td>
<td>22%</td>
</tr>
</tbody>
</table>
Although the presented SK-4PCS performs well in the experiments, there are many aspects that require improvement in future work in order to overcome the limitations of SK-4PCS. The most serious one is that, as already mentioned, a public reference plane is necessary in SK-4PCS in order to interpret the semantic keypoints. In our cases, we used the ground as such a plane. However, the ground may not always be visible in scanning scenes (e.g., it might be blocked). Secondly, to find enough regular geometry features may not always be possible. The semantic information plays a very important role in the proposed registration method such that the lack of semantic information will cause SK-4PCS to fail. We only considered lines, corners, and circles in the presented method to establish the semantic keypoints but that may not always appear in some buildings with sufficient numbers. So, to enrich the feature dictionary of the semantic information is a very useful method to further improve SK-4PCS. The next limitation is the parameter settings. From the tests, we found that SK-4PCS is more sensitive to the global parameter settings than K-4PCS. That is because the features should be detected continually and individually from each cross section. In other words, unreasonable settings may obscure many features from some cross sections, and that will further influence the correspondence searching in later 4PCS transformation calculations.

6. Conclusion

The goals of coarse registration are to align individual scans to a public coordinate system and to provide an initial parameter set for later fine registration. This paper proposed an automatic marker-less coarse registration method, SK-4PCS, for point clouds in building scenes. The main contributions of the proposed method are extending the original 4PCS algorithm to handle heavy datasets and improving K-4PCS in relation to registration accuracy and computational efficiency by using semantic keypoints.

Comprehensive experiments were carried out to evaluate the capability of the SK-4PCS method. We compared SK-4PCS and K-4PCS in relation to the registration accuracy and computational efficiency based on lobby and laboratory datasets. In our tests, the results by SK-4PCS were very encouraging. However, the SK-4PCS scheme is relatively complex in keypoint extraction, and there are some limitations in the presented algorithm. The semantic information is a double-edged sword. It brings the searching process to be more effective and increases the sensitivity of the registration method as well. As the aforementioned, the lack of the semantic information will lead a registration to fail. Thus, it is necessary to enrich the feature dictionary of the semantic information such that to satisfy the fashionable and avant-garde high-tech architecture. To break the most difficult bottleneck in the proposed method i.e. to find a reference plane is also a significant test in the next step. Thus, there remains room for further improvements, e.g., improve the stability of calculations, reduce the parameter sensitivity, and extend the approach to more general cases.

References