CICP: Cluster Iterative Closest Point for Sparse-Dense Point Cloud Registration

Mohamed Lamine Tazir, Tawsif Gokhool, Paul Checchin, Laurent Malaterre, Laurent Trassoudaine

CICP: Cluster Iterative Closest Point for Sparse-Dense Point Cloud Registration

Mohamed Lamine Tazir[a,*], Tawsif Gokhool[b], Paul Checchin[a], Laurent Malaterre[a], Laurent Trassoudaine[a]

[a] *Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, F-63000 Clermont-Ferrand, France;*
[b] *MIS laboratory, Univesité de Picardie Jules Vernes, 80080, Amiens, France*

Abstract

Point cloud registration is an important and fundamental building block of mobile robotics. It forms an integral part of the processes of mapping, localization, object detection and recognition, loop closure and many other applications. Throughout the years, registration has been addressed in different ways, based on local features, global descriptor or object-based. However, all these techniques give meaningful results only if the input data are of the same type and density (resolution). Recently, with the technological revolution of 3D sensors, accurate ones producing dense clouds have appeared as well as others faster, more compatible with real-time applications, producing sparse clouds. Accuracy and speed are two sought-after concepts in every robotic application including those cited above, which involves the simultaneous use of both types of sensors, resulting in sparse-dense (or dense-sparse) point cloud registration. The difficulty of sparse to dense registration lies in the fact that there is no direct correspondence between each point in the two clouds, but rather a point equivalent to a set of points. In this paper, a novel approach that surpasses the notion of density is proposed. Its main idea consists in matching points representing each local surface of source cloud with the points representing the corresponding local surfaces in the target cloud. Experiments and comparisons with state-of-the-art methods show that our approach gives better performance. It handles registration of point clouds of different densities acquired by the same sensor with varied resolution or taken from different sensors.

Keywords: sparse to dense (dense to sparse) registration, density change, cluster, points selection, matching, ICP.

1. Introduction

The problem of dense-sparse registration has received less attention from the scientific community in the past [1]. The majority of research has been focused on dense registration [2, 3], or sparse registration [4, 5, 6, 7]. Recently, the need for sparse to dense registration has come to the limelight, and this is due to the emergence of sensors that produce sparse data like Velodyne [1] LiDAR (Light Detection And Ranging), which is widely used in autonomous vehicles (Google car [8], DARPA Grand Challenge [9]), because of its ability to provide 3D data at a high refresh rate and at a long range [7]. Accurate sensors producing dense clouds also achieved a technological leap with the appearance of 3D laser sensors like Leica P20[2], Riegl VZ400i[3] or Trimble TX8[4], etc. Furthermore, multiple sensors that allow the change of scanning resolution have recently appeared on the market. These sensors can produce point clouds of different densities depending on the chosen resolution. Nevertheless, the difference in cloud density is generally due to the change of the sensor. For example, two different sensors generate two clouds with different point densities, which requires a calibration step between the two sensors in order to exploit the resulting clouds. Calibration is necessary whenever the two sensors are moved, which is a hard and tedious task. On the other hand, the main shortcomings of available point cloud registration methods are their lack of speed due to the increase of input data or their lack of precision due to the decrease in density [7] whereas, for most robotic applications such as localization, these two attributes are highly desired. A recent trend is to use both kinds of sensors [10, 11, 12] to achieve these two sought-after concepts simultaneously, highlighting the importance of dense to sparse registration techniques.

As with all registration methods, an overlapping between the two clouds, usually called source cloud and target cloud, is necessary to determine the rigid transformation between these two clouds perceived from different viewpoints. With the conventional methods that use coherent data from the same sensor, what changes are the representation of points pertaining to the two views. For sensors that also provide intensity or color, these two attributes can be changed if the two clouds are acquired at two different times. Otherwise, except for the noise, nothing else

*Corresponding author
Email address: tazir.med@gmail.com (Mohamed Lamine Tazir)

[1] Velodyne LiDAR: http://velodynelidar.com/

[2] Leica P20: http://leica-geosystems.com/

[3] Riegl VZ400i: http://www.riegl.com/nc/products/terrestrial-scanning/produktdetail/product/scanner/48/

[4] Trimble TX8: http://www.trimble.com/3d-laser-scanning/tx8.aspx

can be changed, neither the number of points nor the spacing between the points. Regarding dense-sparse data, the degree of sampling changes, affecting the number of points and the distance separating these points. In other words, for the same part of the scanning environment, taken from two different viewpoints, the dense cloud will exhibit a larger number of points with a smaller distance separating them, as opposed to the sparser cloud. This change affects all the local characteristics of the points (normals, curvatures), making the conventional methods unsuitable for this type of registration [13, 6].

Recently, Agamennoni et al. [1] addressed the sparse-dense registration issue and proposed a method that improves the standard point-to-point ICP [14, 15] by introducing a probabilistic model for data association. The main idea of this work is to align each point from the sparser cloud with a set of points from the denser cloud. The association with each point is weighted taking into account the uncertainty of the transformation estimate. The problem is formulated as an Expectation-Maximization procedure, during which, weights are calculated throughout the E-step, whilst during the M-step, the rigid transformation is updated from current associations. However, the weakness of this method is that the associations do not change at each iteration. The iterations serve only to optimize the weights. That is why the method should be executed several times, using as input the solution of the previous run, which consumes a lot of time. Moreover, the fact that the point associations do not change at every iteration, makes this method very sensitive to the initial data association.

In this work, we propose a method to align dense and sparse clouds to achieve accuracy and convergence speed. This method surpasses the notion of density by replacing the points sharing the same local surface of the two clouds by a single representative point for the matching step. It is not about sampling, but only for the matching process, the points most likely to match each other are selected. Then, the resulting transformation is used to transform all the source points. The process evolves iteratively in an ICP-like framework, starting with a selection process followed by a pose estimation process. The main contribution of this paper lies in the selection points for the matching process. First, a voxelization is performed on both clouds to maintain the topological details of the scene. Then for each voxel, a normal-based classification of its points is done. Thereafter, only one point of each local surface is maintained for the associating step. As a result, fewer points are used for the matching process, but they are most likely to be associated. Thereby, improving convergence and accuracy simultaneously.

2. Related Work

Registration is a crucial step in several applications, ranging from inspection in the medical domain [16], passing through the detection of objects in computer vision [17], to mapping and localization in mobile robotics [3], which is our main research interest. Registration is located in the front-end of the mapping pipeline [18]. In recent years, the interest and demand for 3D mapping has been greatly increased. This is mainly due to the improvement of acquisition systems on the one hand and the growth of the range of potential applications on the other. Currently, 3D data can be obtained using two technologies: photogrammetry and laser scanning [19]. The laser technology provides direct 3D data, while photogrammetry reconstructs 3D information by techniques such as triangulation from several images of the area under exploration. The advantage of direct 3D data acquisition makes the laser scanner popular for mapping the environment either indoors or outdoors [11]. Moreover, localization can be done at a different timescale compared to the mapping, which requires that the process of localization should be robust to the environment change (such as the lighting change) [11]. In this study, we only focus on laser technology.

Registration algorithms assemble two representations of an environment in a single reference frame. The problem of registration has been dealt with extensively in several studies over the last 25 years. This started with geometric approaches leading to the appearance of the Iterative Closest Point (ICP) algorithm [15, 14]. ICP is used to calculate the optimal transformation fitting two point clouds by a two-step process: matching of points and minimizing a metric describing the misalignment [20]. These two-steps iterate to minimize the matching error and thus improve alignment. In the literature, three groups of registration methods are identified:

- sparse methods (approaches based on features extraction);
- dense methods (approaches exploit all the points in the cloud);
- approaches based on objects.

2.1. Sparse Approaches

Sparse methods are generally used in outdoor environments [12]. They are based on the use of features, which may be points that are easily identified by their apparent character (position, local information contents, mathematical definition, etc.) with respect to the other points. A good feature requires stability and distinctiveness [21]. In other words, detected features should be consistent in all the frames. They should be robust to noise and invariant to rotation, perspective distortion and changes of scale [21, 22, 23]. There are numerous sparse methods in the literature, each one is adapted to specific needs, but all of them share the same workflow. They begin by the identification of the feature estimation model, then the extraction of the set of relevant points (keypoints [24]) corresponding to the feature model. Afterwards, for every point, a local descriptor is computed collecting the shape and appearance of the neighborhood around each point.

Finally, keypoints found in different frames are used to determine correspondences and align the different point clouds.

Among the well-known algorithms is the 3D Scale Invariant Feature Transform (3DSIFT), which is an extension of the 2D version proposed by Lowe in 1999 [25]. The 3D version was adapted by the PCL [26] community using the curvature of points instead of the intensity of pixels [27]. The method uses a pyramidal approach to reach the scale invariance characteristic of features. To achieve invariance against rotation, it assigns orientations to keypoints.

A multitude of methods exploiting 2D information obtained from 3D points have emerged since. Ranging from FAST (Features from Accelerated Segment Test) [28], and going through SURF (Speeded Up Robust Features) [29], until ORB (Oriented FAST and Rotated BRIEF) [30]. These methods, unfortunately, are less robust [23] and are affected by parasitic phenomena such as illumination and weather conditions [21].

Normal Aligned Radial Feature (NARF) [31] is a rotation invariant 3D feature that also operates in range image and has two goals: extract points from stable local surfaces that are near significant changes and from borders. The authors argue that working on range image makes borders explicitly identified by transitions from foreground to background. Indeed, borders usually appear as non-continuous traversals from foreground to background. Still according to the authors, points from stable surface that represent a significant change in a local neighborhood represent robust points that can be detected and observed from different perspectives.

However, all those methods cited above, operate on 2D representation of 3D point cloud. Recently, a method developed by [23] highlights the use of these two strategies together (2D representation and 3D information). It uses a 2D range image, as well as information calculated from 3D points such as normals and curvatures to extract 3D feature point from LiDAR data.

The last category of sparse methods exploits the 3D points directly. Most well-known approaches include the Point Feature Histograms (PFH) which was used in [32] to describe the local geometry around each point, in order to classify them, by means of a multi-dimensional histogram, according to its local nature (flat surface, corner, edge). PFH is a global feature descriptor as it computes a single descriptor for the entire cloud. Fast Point Feature Histograms (FPFH) [33] modifies the mathematical model of FPH in order to reduce its computational complexity. Similar approaches are formed elsewhere, such as VFH (Viewpoint Feature Histogram) [34], CVFH (Clustered Viewpoint Feature Histogram) [35], where features are determined based on the geometric information of 3D points. In the same vein, PCA (Principal Component Analysis) [36] are used in [37] and [38] to establish 3D features for use in recognition and pose estimation.

In any case, sparse methods exploit information about some key points of the scene [21]. They are based on local characteristics of these points, often only geometric characteristics are taken into account [39] although there are other descriptors such as color, intensity, etc. Methods which fall into this category do not require any prior knowledge [40]. Despite this advantage of sparse methods, many of them are not completely adapted to real-time applications [22]. Indeed, features are generally cumbersome to determine, and it is unwise to compute them at each point [22]. Some methods identify a few numbers of locations where their computing may be more efficient [41], but the way these points are determined is time-consuming and hence are often not suitable for the applications that require efficiency. When using sparse methods, another problem occurs which is the necessity of very dense clouds in order to obtain good features, which compromises the use of sparse clouds [1, 21, 42, 4]. More importantly, these methods are environment specific [43], which may result in the rejection of good data [44].

2.2. Dense Approaches

Dense approaches make use of all the points from both clouds, and require an initial guess (transformation) between the two clouds, which makes them sensitive to wrong initialization [41, 40, 22]. Despite the use of all the points, these methods are generally faster than sparse approaches [40].

ICP algorithm belongs to this class of methods. Its strategy consists of supposing an optimistic assumption that there are a number of points in common between the two clouds taken from two different viewpoints. In this way, the algorithm will have an adequate initial estimate of the translation and the rotation, which moves the points of the source cloud to correspond with the points of the target cloud. Applying this assumption, the correspondence of a point will be the closest point to it. In this way, the algorithm will find the closest points of the source cloud in the target cloud. After each iteration, better matches are found, which gradually produce better registration. This is repeated until the convergence of the algorithm is reached. At this stage, the final translation and rotation between the two sets of points are obtained. As pointed out by Pomerleau [3], its easy implementation and simplicity, are both its strength and its weakness. This has led to the emergence of many variants of the original solution, adapted in many ways, throughout the years. Most well-known examples, Chen et al. [14] improved the standard ICP by using point-to-plane metric instead of the Euclidean distance error. This approach takes advantage of surface normal information to reject wrong pairing. However, this approach fails when dealing with clouds of different densities, since normals computation are affected by the change in resolution, presence of noise and distortion [45, 13].

The Normal Distributions Transform (3D-NDT) [46] discretizes the environment in cells, where each one is modelled by matrix representing the probability of occupation

of its points (linear, planar and spherical). Then, a non-linear optimization is performed to calculate the transformation between the two clouds. Nonetheless, according to [45], the NDT is not suitable for systems with low computing power capability.

An efficient approach for dense 3D data registration was presented in [47]. This probabilistic version of ICP called Generalized ICP (GICP) is based on a Maximum Likelihood Estimation (MLE) probabilistic model. It exploits local planar patches in both point clouds which leads to plane-to-plane concept. The authors in that paper show that this algorithm is a generalization of point-to-point and point-to-plane metrics, and the only difference lies in their choice of covariance matrices. Since this algorithm is point-to-plane variant of ICP, it has similar drawbacks, especially those related to normals computation. For instance in [13], it is shown that the non-uniform point densities cause inaccurate estimates, which degrade the performance of the algorithm. Moreover, in [48, 1] the authors affirm that the GICP does not work well in outdoor and unstructured environment.

Serafin et al. [40] extended the GICP algorithm by using the normals in the error function and in the selection of correspondences, which according to the authors, increases the robustness of the registration. NICP [40] works on the projection of the two clouds on range images. For the reference cloud, this range image is recomputed at each iteration, which consumes time. These range images serve primarily for the selection of matched points. The Matched points are selected from the range image, so that they are points that share the same pixel and have compatible normals and curvatures.

Our approach, called CICP for Cluster Iterative Closest Point, uses an (NDT and NICP)-like representation, however, it is different from the NDT in the way it uses the points of each voxel to determine local surfaces and get one representative point from each local surface to the matching process. In contrast, NDT computes a Gaussian distribution in points of each voxel using the vicinity of each point. Whereas, NICP uses an image projection of the voxel grid representation to compute statistics, and considers each point with the local features of the surrounding surface. These features, namely normal and curvature, are calculated for each point from its neighboring points, with a computational complexity of $\mathcal{O}(K \times N)$, where K is the number of the neighboring points used to compute each normal and N the total number of points. Additionally, these features are used later in the process of point matching between the two clouds, as opposed to our method, that does not use normals in the matching process. Because of the difference in density, pattern scanning, and presence of noise, will lead to noisy normals and, hence, inaccurate results.

2.3. Approaches based on Objects

Object-based methods have chosen to take advantage of higher-level representations, including 3D objects (solid shapes), 2D forms (plans), or 1D (segments). This concept allows them to benefit from a massive compression of information [49].

In [50], the authors propose a SLAM algorithm that combines recognition and 3D reconstruction of maps at the level of the object. During the navigation process, the algorithm uses prior knowledge of specific objects that are supposed to be in the environment, to perform a recognition task. These objects are used as top-level features to optimize the ICP-based pose refinement. However, this work is limited to the indoor environment and the specific known objects.

Fernandez-Moral et al. [51] propose a registration method based on planar surface. This paper represents an extension of the work published in [52] which deals with the recognition places in indoor environments by extraction of planes. The extension is mainly focused on adding a probabilistic framework to account for the uncertainty model of the sensor. Whereas for [49], this approach is applicable only to small and indoor environments. The method uses the region growing technique [53] to obtain the planar patches from the scene and represents them using a graph. Other techniques may also be used such as RANSAC [54] and Hough transform [55] as in [26] and [7]. However, for our proposal (dense-sparse registration), sparsity poses a real problem to get accurate segmentation [7].

The segments are also used in the process of matching. In [4], the authors introduce a Velodyne point cloud registration method based on line clouds. The algorithm starts by sampling the two clouds into sets of random segments, then the correspondence is made by a strategy similar to the ICP between the two sets of lines. Dubé et al. [49] use a segment-based method for a loop-closure purpose. This method has the advantage of compressing the point cloud into a group of distinct elements, which reduces false matches and optimizes the time required for correspondence.

Object based methods suffer from imperfect segmentation [49]. Their matching tends to reject a lot of potentially useful data, since they exploit information belonging to some simplistic geometric models [44].

CICP differs from these three sub-categories in the nature of its data and how these data are used. As input, it takes point clouds of different resolution, gathered by different sensors, or with the same sensor. It is based only on the geometric information of points, which makes it independent of weather and illumination conditions. The proposed approach aims to cluster points of the same surface as one topological pattern, and replace all the points held by this model by one representing point for the matching step. The main algorithm is based on point-to-point matching alignment.

Table 1 summarizes the main differences between the proposed method and the three sub-categories identified in the prior related work.

Table 1: Main differences between CICP and the state-of-the-art methods.

	Dense methods	Sparse methods	Object-based methods	CICP
No prior information required	x [46] [40]	✓ [13] [56]	✓ [57]	✓
Use all points	✓ [40]	x [34] [35]	x [4]	✓
No risk of loss of good data	✓ [14] [15]	x [58] [59]	x [7]	✓
Based on geometric information	✓ [41]	✓ [23] [32] [35]	✓ [21]	✓
Use of normals in correspondences choice	✓ [47] [14] [40]	✓ [37] [56]	✓ [51]	✓
Compress point cloud in a set of distinct elements	not concerned	x [33]	✓ [49] [52] [60]	✓
Not environment specific	✓ [46]	x [43]	x [50] [51]	✓
Not affected by imperfect segmentation	not concerned	not concerned	x [52] [61]	✓
Does not require a very dense cloud	✓ [6]	x [62]	✓ [21]	✓

3. Our Contributions

In this paper, a novel approach for sparse to dense (or dense to sparse) registration is introduced, exploiting normals differently. The desired contributions are as follows:

1. A new selection strategy is proposed by keeping only points which are most likely to be associated in the matching phase, thereby improving on convergence and accuracy simultaneously.
2. The proposed method is totally independent of the density (amount of points, scanning resolution) of the two clouds, scanning patterns (nature of sensors). It takes as input point clouds of different resolution, gathered by different sensors, or with the same sensor.
3. A novel mathematical definition of sparse and dense concept is proposed to accomplish these objectives.

All the considerations outlined in this section will be demonstrated in the results section and these claims are further consolidated in the Discussion section (cf. Section 7).

4. General Formulation

4.1. Mathematical Definition

Dense and sparse are terms used to describe the state of points within a cloud. This includes their quantity, distribution and resolution. The distinction between these two terms is rather vague, and depends on the context.

Suppose we have two clouds of the same environment, with the same dimensions and taken from the same viewpoint, they will probably have been taken by different sensors or by the same sensor with varied resolutions. Let us assume that there is a large degree of variation between their densities. The dense cloud will exhibit a larger number of points with a smaller distance separating them, as opposite to the sparser cloud. The concept of density in 3D is always linked to a given volume. To get the same volumes, the two clouds are divided into voxels (subdivisions) of the same size. At this stage, the main clouds integral characteristics are the set of voxels V and the set of points P. The relation between these two sets determines whether the cloud is sparse or dense.

Below, we give some basic definitions, and we introduce the "voxelic density" definition in theoretical and practical cases:

Definition 1 (Voxel in $\mathbb{R}^3$). A voxel v with center $\omega = (x_0, y_0, z_0)$ and rayon r, is a set of points $P(x, y, z)$ if:
$v = \{(x, y, z) : \max\{|x - x_0|, |y - y_0|, |z - z_0|\} <= r\}$

Definition 2 (Set of all voxels in P). Let v a voxel of V_P, we say that V_P is a set of all voxels in P if:
$V_P = \{\forall v \in V_p, v \subset P\}$

Theoretically, a dense cloud is:

Definition 3 (Voxelic density). P is dense $\leftrightarrow$
$\forall v \in V_p, \exists p \in P : p \in v$

According to this last definition, a point cloud is called dense, if and only if, there is always at least one point belonging to a voxel, whatever the voxel size.

Unfortunately, for practical reasons, this definition is not always verified. Because of this, we propose definitions 4 and 5:

Definition 4 (Sparse Cloud). A sparse cloud is a cloud $C = (V, P)$ in which:
$|P| = \mathcal{O}(|V|)$.

Definition 5 (Dense Cloud). A dense cloud is a cloud $C = (V, P)$ in which:
$|P| = \mathcal{O}(k.|V|)$, with $k > 2$.

$\mathcal{O}$: proportionality operator.

Definitions 4 and 5 are proposed to frame the notions of sparsity and density of point clouds. The voxel size is set according to the number of points in the sparse cloud, so that each voxel contains at least one point. This choice ensures a significant difference in density between the two

clouds. A dense cloud, in our case, contains at least twice as many points as the sparse cloud. Otherwise, they are considered as equivalent.

4.2. Iterative Closest Point: The Algorithm

Our proposed algorithm, named CICP in short for cluster iterative closest point, adopts the general scheme of the ICP algorithm. For this reason, we will discuss here the most important items of that algorithm. As it is well-known, the ICP algorithm is an iterative registration method, which consists in putting the points of the source cloud into the frame of the target cloud in order to generate a unique and consistent point cloud. To do this, a translation and a rotation, which make the points of the source cloud move to correspond with the points of the target cloud must be found by the algorithm. Moreover, this algorithm must identify the points of the two clouds that correspond to each other. The strategy of the ICP algorithm consists in taking an optimistic assumption that there are a number of points in common between the two clouds taken from two different points of view. In this way, the algorithm will have a good initial estimate of the rotation $\mathbf{R}$ and the translation $\mathbf{t}$. Applying this assumption, the correspondence of a point will be the closest point to it. In this way, the algorithm will find the closest points of all source points corresponding to the points of the target cloud. Once it has these correspondences, it can improve the estimate of $\mathbf{R}$ and $\mathbf{t}$, by solving this optimization:

$$\mathbf{R}, \mathbf{t} = argmin_{R_j, t_j} \sum_{i=1}^{N} \|d(p_i, q_i)\|_2 \tag{1}$$

where p and q denote the pairs of corresponding points in the two clouds and d represents the distance separating the points of each pair.

After each iteration, better matches are found producing progressively better registration. This is repeated until the algorithm converges. It converges when the distance d is less than a certain threshold. Once this convergence is reached, the final rotation and the translation between the two sets of points are obtained. Rusinkiewicz [63] identify six distinct stages in this algorithm:

1. **Selection**: selection of a set of points from one or both clouds of input points. This first stage seeks to reduce the number of points of the input clouds by applying one or more filters [64]. The way in which these points are selected has a direct impact on the convergence of the algorithm, and especially on the computation time necessary for convergence, in particular, when handling very dense datasets. In this stage, we can find strategies like uniform [65] or random sampling [66], sampling according to the orientation of normals [63], statistical sampling [67] and outliers filtering [22]. However, as we deal with sparse and dense clouds, classical selection strategies can bring improvements for dense point clouds, but for sparse point clouds, they might cause further degradation to the characteristics and information carried by these points. For this reason, we chose to use all the points in the two clouds without making any modification on points of both clouds.
2. **Matching**: this step represents the key operation in the ICP algorithm. It consists in coupling corresponding points from both clouds. These correspondences are obtained by seeking, for each point of the source cloud, the nearest point in the target cloud. The definition of the "nearest point" determines the matching technique used [68]. Several techniques of nearest-neighbor-search (NNS) are used to optimize the time of this step, as it used to be the most demanding step in terms of computation time [68]. The authors of [69] and [68] assert that "k-d trees" is the best technique to find the nearest neighbor. For this reason, we use it in our implementation.
3. **Weighting**: assignment of weights to matched pairs of points. It aims to strengthen the contribution of correspondences believed to be correct and mitigate the effect of false matches [13].
4. **Rejection**: reject the pairs of points that do not contribute positively to the convergence of the algorithm, such as outliers, occluded points (points that are not visible in one of the acquisitions) or unpaired points (points of one cloud that do not find correspondents in the second cloud).
5. **Error metrics**: it defines the objective function which is minimized at each iteration of the algorithm. Three metrics are commonly used: point-to-point [15], point-to-plane [14] and plane-to-plane [47].
6. **Minimization**: minimize the error metric to bring the points of the source cloud and align them with the points of the target cloud.

The algorithm terminates when a maximum number of iterations is reached or a variation relative to the error metric is reached. In many cases, the algorithm converges quickly but not necessarily towards the optimal solution. Several problems may arise, namely:

- noises and outliers that can cause biased results,
- partial overlap.

5. Proposed Method

This work proposes a novel approach that deals with dense-sparse registration. We adopt the Rusinkiewicz decomposition and propose a new selection strategy, which aims to improve the pairing process and make it reliable for the purpose of this paper. Figure 1 illustrates the workflow of the proposed method.

CICP starts with the estimation of normals of the two clouds. Then, it takes the target cloud first and subdivides it into small voxels. The points of each voxel are

subjected to a classification process based on their normals, giving rise to different groups of points, according to the geometric variation of each voxel. Each group of points represents a local surface since they share the same normal vector. Next, from the points of each small local surface, a single point is chosen to represent this surface during the matching process. In our case, we take the closest point to centroid of each local surface. Regarding the source point cloud, its points are transformed with their normals by the initial guess of the relative transformation. Then, this cloud undergoes the same steps as the target cloud: voxelization, normals-based classification, designation of points representatives. At the end of these steps, the method comes up with two sets of points from the two clouds. Each set contains the most probable points to match with the points of the second set (this is specifically in the overlapping area of the two clouds, as it reflects the same geometry seen from two different viewpoints).

5.1. Selection

The main contribution of this paper is the proposal of a new selection strategy. As mentioned above, instead of matching point-to-point as the classical ICP variants, points pass through an election process, which gives rise to one representative point for each small region. These representatives appear as the most likely points to be matched between each other. These good matches ultimately result in an accurate motion between the two clouds. This election process is based on 3D position of points and their normals. It consists of three sub-tasks: (1) voxelization, (2) clustering and (3) Representative election. The first task performs a spatial grouping which attempts to preserve the topological information based on the 3D position of the points. A set of 3D cubic regions (voxels) is generated where all points within the voxel have very close spatial positions. The second task bundles all points of each voxel based on their normals. Once this grouping is done, we perform the last task, which selects one point for each cluster (local surface) in the voxel for the matching process. But before that, normals need to be calculated.

5.1.1. Normal Estimation

Normal segmentation of geometric range data has been a common practice integrated in the building blocks of point cloud registration. Most well-known point to plane and plane to plane state-of-the-art registration techniques make use of normal features to ensure a better alignment. However, the latter is influenced by noise, pattern scanning and difference in densities. Consequently, the resulting normals in both a source point cloud and a target point cloud will not be perfectly adapted, thereby influencing the alignment process, due to weak inter-surface correspondences. In order to support these claims, an illustration of sparse to dense registration is given in Figure 2. A dense point cloud is obtained from a 3D LiDAR Leica P20 scanner whilst the sparser one is extracted from an HDL-32E Velodyne. Figures 2 (c), (d), (e), (f) are samples of various places in a scene. The 3D points of the source and target clouds are represented in blue and green respectively, whilst their normals are in white and red. These figures depict the dissimilarity between normals pertaining to the same surface, which theoretically should have the same orientations. This change is due to the different disturbances mentioned above. This is the major problem of the methods that use geometric features according to [21, 13]. Additionally, according to the authors of [70], the calculation of normals on a large dataset is computationally expensive.

To overcome this problem, we use normals only to distinguish the different local surfaces (group each surface alone). For the rest of the algorithm, we use x, y and z coordinates of each point without having recourse to their normals. In other words, we use normals only to distinguish the different surfaces, but we do not use them in the alignment process.

Normals are computed once for each cloud at the beginning of the algorithm. Source normals are transformed at each iteration by the transformation found. We use Principal Component Analysis (PCA) [36] to determine normals vectors of point cloud, as it is the most performant method used to compute normals according to [68] and [71]. PCA-based algorithm is usually used to analyze the variation of points in the three directions. Normal vector corresponds to the direction with minimum variation. We can also imagine the use of the dominant directions of the points as a characteristic of designation of the cluster within each voxel, instead of normals. We have chosen to use normals, as they are classical and common features.

From the eigen decomposition of the covariance matrix of considered nearest neighbors, the eigenvector corresponding to the minimum eigenvalue represents the normal vector. The covariance matrix can be calculated from the following equation:

$$\mathbf{C} = \frac{1}{k}\Sigma_{i=1}^{k}(p_i - \bar{p})^T(p_i - \bar{p}) \tag{2}$$

where k is the number of considered nearest neighbors; p_i, $i = 1 : k$ are kNN points and $\bar{p}$ is the mean of all k neighbors.

Algorithm 1 shows how to calculate the normals with the PCA method.

5.1.2. Voxelization

The voxelization is applied in order to maintain the topological details of the scrutinized surface. As normal computation depends on the number of neighbouring points and as the resolution of points of the two clouds is different, voxelization with the same voxel size aims to generate equivalent local regions in the two clouds. A common criterion of comparison now becomes feasible. Therefore, the voxel size parameter is of paramount importance for our technique and it should be chosen carefully in order to keep

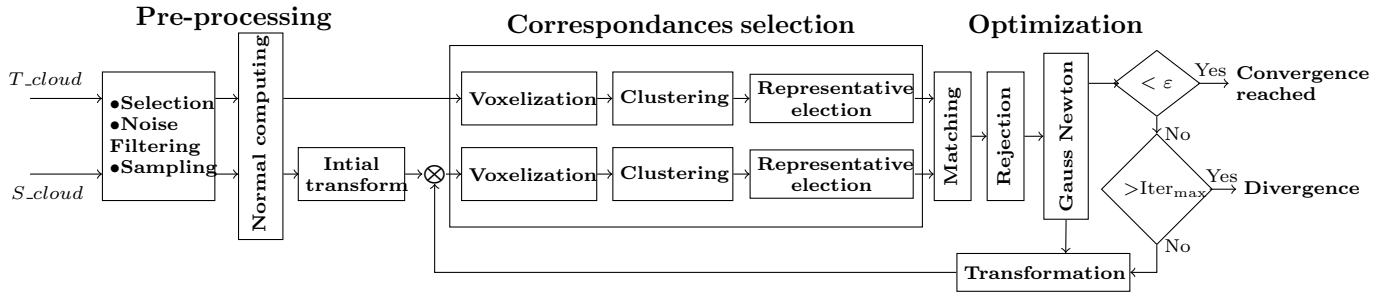
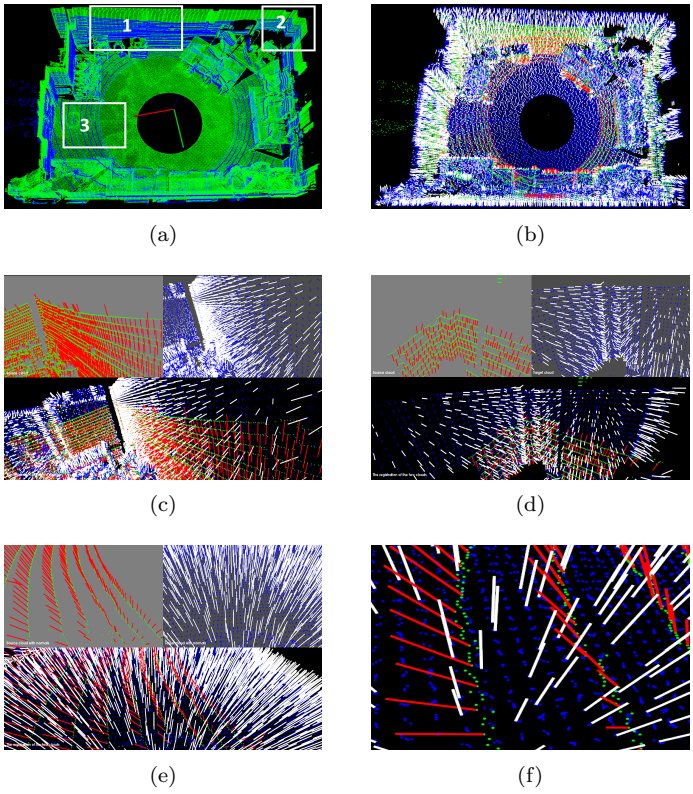


Figure 1: Overview of the CICP pipeline. Given two point clouds, CICP starts by computing the surface normals of the two clouds. It looks for points sharing the same local properties, and then elects one representative point from each local cluster. This election process is based on 3D position of points and their normals. It consists of three sub-tasks: (1) Voxelization: a set of 3D cubic regions (voxels) is generated where all voxel points have very close spatial positions. (2) Clustering: classify all points of each voxel according to their normals. (3) Representative election: once this grouping step is completed, the last task consists in selecting one point from each cluster (local surface) in each voxel. Representative points serve as candidates for correspondence process. As a result, few points are used in the matching process, but which are most likely to be associated, thereby, improving on convergence and accuracy simultaneously.



(a) (b) (c) (d) (e) (f)

Figure 2: Dense to sparse registration: (a) registration of a dense point cloud obtained from the Leica P20 LiDAR with 88 556 380 points and a sparse point cloud obtained from an HDL-32E Velodyne with 69 984 points using our proposed method; (b) normal vectors corresponding to (a); (c), (d) and (e) are exploded views of places indicated in (a); (f) is a close up view of (e).

Algorithm 1: Compute normals with PCA.

Input: pointXYZ P, num_neighbors
Output: vector normals
Initialize: vector normal, vector neighbors, vector $\bar{P}$, matrix Q, H, U, V
begin
  foreach *point p in P* **do**
    // Extract the neighbors
    $neighbors = nearestKSearch(p, num\_neighbors, P)$
    // Calculate the centroid of neighbors
    $\bar{P} = sum(neighbors)/num\_neighbors$
    // Compute the covariance matrix
    $Q = (neighbors - \bar{P})$
    $H = Q.transpose() * Q$
    // Compute the eigenvectors
    [U, V] = $svd\_decomposition(H)$
    // Sort the eigenvectors by decreasing eigenvalues
    $U = sort\_decrease(U)$
    // Extract the normal
    normal [0] = $U(0, 2)$
    normal [1] = $U(1, 2)$
    normal [2] = $U(2, 2)$
    // Stack normal in container
    $normals.emplace\_back(normal[0], normal[1], normal[2])$
  end
  return *normals*
end

the fundamental characteristics of both point clouds; be it dense or sparse with topological details. In our case, the set of rules mentioned previously in the Section 4.1 must be verified.

At the beginning, the procedure applies a bounding box to the entire sparse cloud by finding the minimum and maximum positions of points along the three axes X, Y and Z. The number of voxels for this bounding box is determined by the number of points and the voxel size is deduced. The same procedure is applied to the dense cloud. For more details, see voxelized point clouds in Algorithm 2.

1. Voxel assignment: each voxel is identified by a unique

Algorithm 2: Voxelization of a point cloud.

Input: pointXYZ P, voxelSize
Output: vector voxels
Initialize: minX = maxX = $P[0].x$, minY= maxY= $P[0].y$, minZ = maxZ = $P[0].z$, numDivX= numDivY =0
begin
  // Create a bounding box for all the points of P
  foreach *point p in P* **do**
    if $(p.x < minX)$ **then** $minX = p.x$;
    if $(p.x > maxX)$ **then** $maxX = p.x$;
    if $(p.y < minY)$ **then** $minY = p.y$;
    if $(p.y > maxY)$ **then** $maxY = p.y$;
    if $(p.z < minZ)$ **then** $minZ = p.z$;
    if $(p.z > maxZ)$ **then** $maxZ = p.z$;
  end
  // Calculate number of voxels along each axis
  numDivX = $(maxX - minX)/voxelSize + 1$
  numDivY = $(maxY - minY)/voxelSize + 1$
  numDivZ = $(maxZ - minZ)/voxelSize + 1$
  // Assign each point p to its corresponding voxel
  foreach *point p in P* **do**
    $i = (p.x - minX)/voxelSize$
    $j = (p.y - minY)/voxelSize$
    $k = (p.z - minZ)/voxelSize$
    $idx = i + j \times numDivX + k \times numDivX \times numDivY$ (3)
    voxels $[idx].push\_bach(p)$
  end
  // Erase empty voxels
  voxel.erase $(remove\_if(voxels.begin, voxels.end, container\_is\_empty), voxels.end)$
  return *voxels*
end

linear index. If i, j, k represent the voxel indices in the X, Y, Z dimensions, respectively, $numDivX$, $numDivY$ are the number of voxels along X and Y axes, the formula to encode the linear index [72] is:

$$idx = i + j \times numDivX + k \times numDivX \times numDivY \quad (3)$$

According to (3), we assign an index idx to each point. This relationship allows direct access to the desired voxel, thereby avoiding a linear search [73].

2. Voxels suppression: as the shape of the point cloud is arbitrary, the step of delimiting points by a bounding box creates many empty voxels which are later pruned out. Eventually, voxelization helps to filter noise from voxels where there is insufficient occupational evidence.

5.1.3. Clustering

The process of electing one point from each local surface makes them good candidates for point correspondence searching, thereby rejecting wrong matches impacting alignment accuracy. At first, all the "voxelized" points are taken and a classification method is applied to identify points belonging to the same surface. In our work, k-means clustering [74] is used as the classification technique based on the normal of each point. The appropriate number of clusters (local surfaces) within each voxel is determined using the Elbow method [72, 75]. An illustration of the described approach is given in Figure 3.

Grouping the point clouds using their normal aims at:

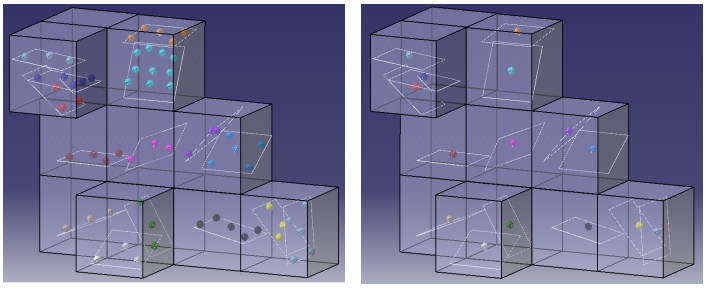
- improving the robustness of the matching step by only allowing the pairing of compatible points,
- reducing the amount of data to be processed during the matching stage.

5.2. Matching

The clustering process generates a reduced, but different number of points in both clouds. These two sets of points are used for matching. Based on our bibliographic research, the best technique identified is the k-d trees (Section 4.2). We use the k-d trees implemented in PCL [26] directly, which is based on the FLANN library [13]. The correspondence is obtained by finding, for each point of the source cloud, the nearest point in the target cloud. This is accomplished using L_2 norm. Since the number of points used for matching is different in the two pairing sets, there will be some wrong correspondences. This is handled in the rejection step, which aims to reduce these false matches.

5.3. Weighting

The aim of weighting is to reduce the influence of outliers on the alignment process. We tested two weighting strategies: Huber weighting [76] and Tukey weighting [77]. However, the tests showed a minimal influence of these strategies on our data. Consequently, on our implementation, we do not use any weighting strategy. This is the



(a) Voxelized and clusterized point cloud (b) Electing one point from each cluster for the matching phase

Figure 3: Voxelized/normal-based clustering for matching process.

same conclusion as found in [68], which affirms that the weighting stage might be removed from the ICP algorithm.

5.4. Rejection

We opted for the distance-based rejection [39, 22], as it is the most basic way to eliminate the wrong pairings. This simple and powerful strategy consists in eliminating the correspondences which have distances greater than a given threshold [65, 22]. This aims to reject the pairs of points that do not contribute positively to the convergence of the algorithm, such as unpaired points (as the number of points used for matching is different in the two pairing sets) and outliers.

5.5. Error metrics

After the matching step, which results in a selection of an equivalent number of representative points in both clouds, the three metrics commonly exploited in the literature, namely point-to-point, point to plane and plane to plane, can be used. However, for the sake of simplicity, we use the point-to-point metric:

$$\mathbf{E} = \sum_{i=1}^{N} \|\mathbf{R}p_i + \vec{\mathbf{t}} - q_i\|^2 \tag{4}$$

5.6. Optimization

The optimization is used to determine the transformation from the set of finding pairs. Given a set of correspondences, rotation and translation between the two frames are calculated using Gauss-Newton iterative least square algorithm [78] (Algorithm 3).

In the case of the point-to-point metric, the error function to be minimized is given by:

$$\mathbf{E}\left(x\right) = \sum_{i=1}^{N} \|\vec{\mathbf{T}}\left(\tilde{x}\right)p_i - q_i\|^2 \tag{5}$$

where $\vec{\mathbf{T}}(\tilde{x})$ defines the displacement between the points of the source cloud p_i and the points of the target cloud q_i, and x a vector which belongs to R^6, representing linear velocities $\vartheta = [\vartheta_x\vartheta_y\vartheta_z]$ and angular velocities $\omega = [\omega_x\omega_y\omega_z]$.

Suppose now that only an approximation $\hat{\mathbf{T}}$ of $\vec{\mathbf{T}}\left(\tilde{x}\right)$ is known. In this case, the registration problem consists in finding the incremental transformation $\mathbf{T}(\mathbf{x})$:

$$\vec{\mathbf{T}}(\tilde{x}) = \hat{\mathbf{T}}\mathbf{T}(x) \tag{6}$$

Such that the differences between the positions of the source points registered by the transformation $\hat{\mathbf{T}}\mathbf{T}(x)$ and those of the target cloud, are zero.

$$\mathbf{E}\left(x\right) = \sum_{i=1}^{N} \|\vec{\mathbf{T}}\left(\tilde{x}\right)p_i - q_i\|^2 = 0 \tag{7}$$

$\mathbf{E}(x)$ is the vector of dimensions $(m \times n) \times 1$ containing the errors associated to each point.

Since an approximation of the displacement $\vec{\mathbf{T}}(\tilde{x})$ is known, the increment $\mathbf{T}(x)$ is assumed to be small. In this case, it is possible to linearize the vector $\mathbf{E}(x)$ by performing a Taylor series approximation around $x = 0$:

$$e(x) = e(0) + \mathbf{J}(0)x + \frac{1}{2}\mathbf{H}(0,x)x + O(\|x\|^3) \tag{8}$$

where $\mathbf{J}$ is the Jacobian matrix of the error vector $\mathbf{E}$, with dimensions $(m \times n) \times 6$ and represents the variation of $e(x)$ as a function of each component of x:

$$\mathbf{J}(x) = \nabla_x e(x) \tag{9}$$

and the matrix $\mathbf{H}(x_1, x_2)$ of dimensions $(m \times n) \times 6$, is defined $\forall (x_1, x_2) \in \mathbb{R}^6 \times \mathbb{R}^6$ by:

$$\mathbf{H}(x_1,x_2) = \nabla_{x_1}(\mathbf{J}(x_1)x_2) = \left[\frac{\partial^2 e_1(x_1)}{\partial {x_1}^2}x_2 \frac{\partial^2 e_2(x_1)}{\partial {x_1}^2}x_2 \dots \frac{\partial^2 e_n(x_1)}{\partial {x_1}^2}x_2\right]^T \tag{10}$$

where each Hessian matrix $\frac{\partial^2 e_1(x_1)}{\partial {x_1}^2}x_2$ represents the second derivative of $\mathbf{E}$ with respect to x.

The system of equations (2) can be solved with a least-squares method. This is equivalent to minimizing the following cost function:

$$O(x) = \frac{1}{2}\|e(0) + \mathbf{J}(0)x + \frac{1}{2}\mathbf{H}(0,x)x\|^2 \tag{11}$$

A necessary condition for the vector x to be a minimum of the cost function is that the derivative of $O(x)$ is zero at the solution, i.e. $x = \tilde{x}$:

$$\nabla_x O(x)|_{x=\tilde{x}} = 0; \tag{12}$$

In this case, the derivative of the cost function can be written:

$$\nabla_x O(x) = \left(\mathbf{J}(0) + \mathbf{H}(0,x)\right)^T \left(e(0) + \mathbf{J}(0)x + O(\|x\|^2)\right) \tag{13}$$

The standard method for solving (12) is Newton's method. It consists of incrementally determining a solution x by:

$$x = -\mathbf{Q}^{-1}\mathbf{J}(0)^T e; \tag{14}$$

where the matrix $\mathbf{Q}$ is written:

$$\mathbf{Q} = \mathbf{J}(0)^T\mathbf{J}(0) + \sum_{i=0}^{N} \frac{\partial^2 e_1(x_1)}{\partial {x_1}^2}|_{x=0} e_i \tag{15}$$

However, Newton's method requires the calculation of the Hessian matrices, which is expensive in computation time. Nevertheless, it is possible to approximate the matrix $\mathbf{Q}$ with a first order approximation by the Gauss-Newton method:

$$\mathbf{Q} = \mathbf{J}(0)^T\mathbf{J}(0) \tag{16}$$

For this genre of non-linear optimization problem, the Gauss-Newton method is preferred, because, on the one

hand, it makes it possible to ensure a definite positive matrix $\mathbf{Q}$ and, on the other hand, to avoid the rather expensive calculation of the Hessian matrices.

Under these conditions, at each iteration, a new error $\mathbf{E}$ and a new Jacobian matrix $\mathbf{J}(0)$ are computed in order to obtain the new value of x by:

$$\mathbf{x} = -\Big(\mathbf{J}(0)^T\mathbf{J}(0)\Big)^{-1}\mathbf{J}(0)^T e(x) \tag{17}$$

and to update the rigid transformation by:

$$\hat{\mathbf{T}} \longleftarrow \hat{\mathbf{T}}\mathbf{T}(x) \tag{18}$$

In general, the minimization is stopped when the error: $\| e \|^2 < \alpha$ occurs, or when the calculated increment becomes too small: $\|x\|^2 < \varepsilon$, where α and ε are predefined stop criteria.

Algorithm 3: CICP Algorithm.

Input: targetCloud, sourceCloud; voxelSeize, $\hat{T}$
Output: Optimal T
Intialize: NormalXYZ T_normals, S_normals; PointXYZ T_match, S_match
begin
  T_normals = normalComputing (targetCloud)
  S_normals = normalComputing (sourceCloud)
  T_match = normalClustering (targetCloud, T_normals, voxelSeize)
  while ($iteration < iter\_\max \| |x| > \varepsilon$) **do**
    sourceCloud = transform (sourceCloud, S_normals, $\hat{T}$)
    S_match = normalClustering (sourceCloud, S_normals, voxelSeize)
    EstablishCorrespondences (T_match, S_match)
    distanceRejection (distThreshold)
    compute the Jacobian $\mathbf{J}$ (9)
    compute the error vector $e(x)$ (8)
    compute the increment x (17)
    update the pose $\mathbf{T}$ (18)
    iteration ← iteration + 1
  end
  return T
end

5.7. Analysis of the cost function

In this section, a more in-depth analysis of the cost function is carried out. The cost function is based on a sum of squared error (SSE) term as shown by equation (5). This is an undeniable problem considering the fact that for a non-linear optimization problem (as is our case), whose domain is non-convex, it may contain several local minima. The local convexity of the SSE estimator around the solution is impacted by several factors; sensor observability, sensor noise, uncertainties induced whilst taking measurements. Therefore, a mathematical condition for convergence is generally difficult to establish.

However, the optimization domain can be sampled to provide a qualitative analysis of the convexity of the estimator. This is illustrated in Figure 4 where the root-mean-square error ($RMSE$) (in meters) is shown versus two groups; translational and rotational couples. It is observed that for a typically chosen subsampled point cloud set (dense: 926 725 points, sparse: 71 584 points), the estimator is convex for the translation as inferred from its formulation and hence, the result is not directly concerned by the initialization of the algorithm. However, the minimiser, though it exhibits a globally convex profile, contains one or several local minima along the way. This implies that initial values for the relative rotation must be carefully given locally around the solution.

6. Results

We implement our CICP approach in C++ without code optimization, and we conduct multiple experiments to evaluate it. Two different data sets are used: (1) point clouds acquired by different sensors; (2) point clouds generated by a single sensor by varying the scan resolution. These two datasets are carried out on indoor and outdoor environments. The indoor scene is represented by a typical office environment, which is symbolized with walls, desks and chairs. And the outdoor environment (PAVIN [5]) is an experimental site for the development of automated vehicles in realistic urban environment. These different datasets provide a good platform to investigate the performance of the proposed method. We first show its results for the registration of two sparse and dense clouds, acquired with two different sensors, and enumerating different indoor and outdoor environments (Section 6.1). Then, we compare our results with the existing sparse and dense methods (Section 6.2). Thereafter, we perform registrations between multiple clouds from the same sensor, but with different resolutions (Section 6.3). Finally, registrations between clouds from different sensors are carried out (Section 6.5).

The computational efficiency of the algorithm is beyond the scope of this paper. We would rather focus on the methodology.

The experimental is set up as shown in Figure 5. The center of the two sensors; Velodyne HDL32 and that of the Leica P20 are perfectly superimposed with the help of the STANLEY Cubix cross line laser. The velodyne is then physically displaced and rotated by known translations and rotations from the graduated set up in order to perturb the 6 degrees of freedom transformation. Data

[5] PAVIN: http://www.institutpascal.uca.fr/index.php/en/the-institut-pascal/equipments

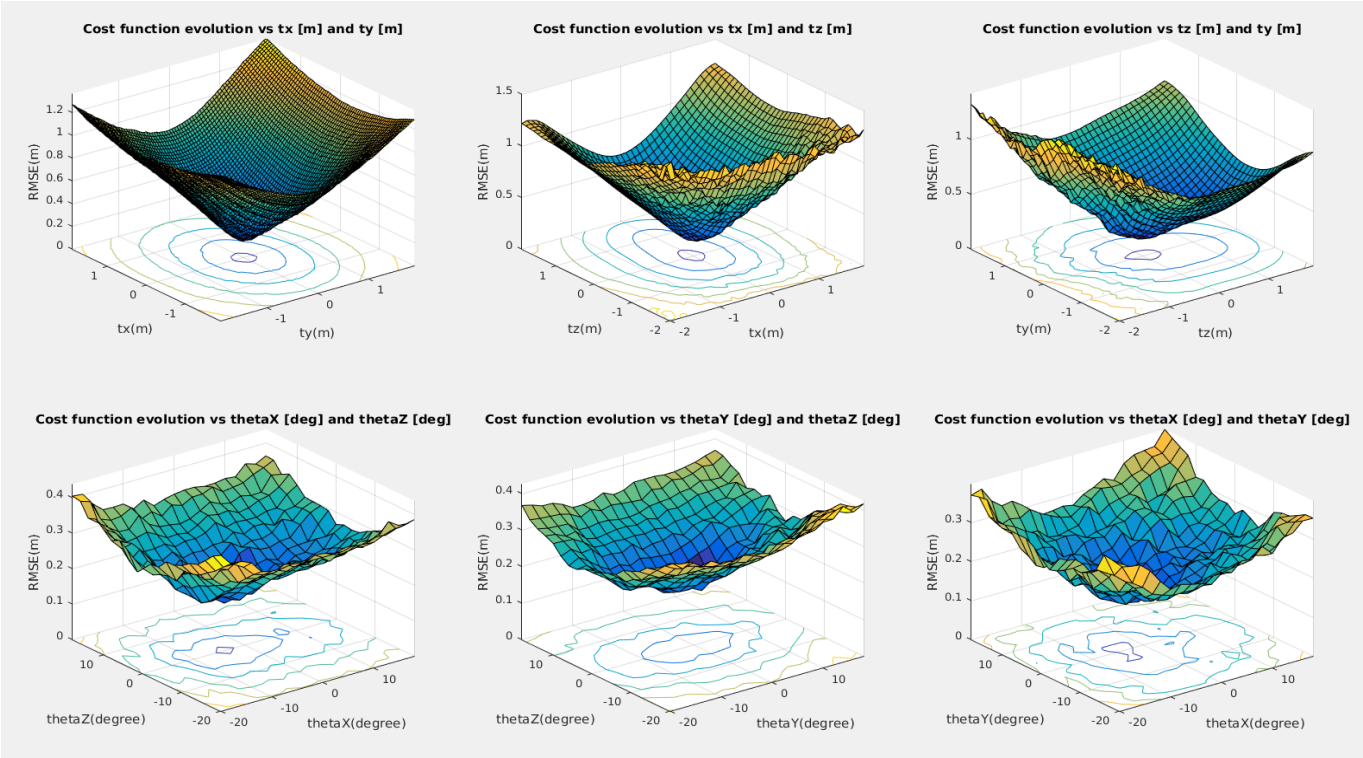


Figure 4: Convergence domain for the office scene. First row represents the $RMSE$ with respect to the three possible DoFs in translation. $t_x, t_y, t_z \in [-2\text{ m}, 2\text{ m}]$ with a descretization of 10 cm. The second row shows the rotation domain where each DoF takes values in $[-20°, 20°]$ with a step of 2°.

acquisition is then performed under different scenarios in order to test our CICP algorithm. Table 3 below summarizes the various experiments performed in a controlled environment. For each experiment, CICP is initialized at Identity, i.e. $x = [0, 0, 0, 0, 0, 0]$.

6.1. Dense-Sparse Registration with CICP

The purpose of this first experiment is to evaluate our CICP method. For that, we choose two clouds acquired with different sensors; the denser cloud produced by a 3D LiDAR Leica P20 laser scanner and the sparser cloud with an HDL-32E Velodyne LiDAR sensor. A Leica P20 generates very detailed and dense point clouds as shown in Figures 6(a), 6(c). Depending on the resolution chosen during the scanning process, these clouds can exceed 100 millions of points for a single scan. For reasons of compatibility with the available computational equipment, which provided an Intel Core i74800MQ processor, 2.7 GHz, and 32 GB of RAM, we perform a sampling process using method described in [72] in order to reduce the number of points to the order of few millions without losing useful information. By compressing data, we provide a more compact 3D representation of point clouds whilst maintaining the notion of density and without affecting the initial structure of the scanned subject. Figures 6(b), 6(d) illustrate the output of the sampling process with 986 344 and 2 732 783 points for the office and PAVIN scenes, respectively. As for the Velodyne HDL-32E, this sensor generates sparse point clouds that do not exceed 70 000 points. This represents a ratio of 14 times between the two clouds from the first environment and a ratio of 40 times, for clouds of the second environment.

Figure 7 shows the registration process of such point clouds using the CICP method. On the left, the green cloud is from the LiDAR Leica P20 and the blue cloud is from the Velodyne HDL32-E. The corresponding results are shown on the right. Table 2 includes the various parameters that manage the registration. The voxel size is set according to the number of points in the sparse cloud in order to verify the definition proposed in Section 4.1. In order to optimize the computing time, it is better to choose the sparse cloud as the source cloud, since the latter is transformed and clustered at each iteration.

Table 2: CICP configuration parameters.

Parameter	Value
Voxel size	8 cm
Rejection distance	$0.2 - 0.5$ m
Max iteration	500
Translation tolerance	10^{-3} m
Rotation tolerance	10^{-4} °

In order to verify the convergence of the optimization, a comparison between the two clouds at the start and at the end of the registration process is recorded together with the convergence profile obtained. It is summarized in the

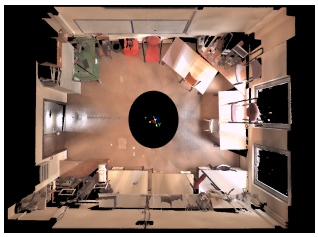
Figure 5: Experimental set up for data collection from Velodyne HDL32 (left) and Leica P20(right) sensors.

Table 3: CICP registration applied to mainly two compiled datasets; OFFICE and PAVIN. The resolutions of corresponding dense and sparse cloud are given along with the initial physical measured transformation from our set up given by the first row of each experiment, whilst the second row depicts the results output by our algorithm. Convergence is evaluated from the $RMSE$ and the number of iterations required for full registration.

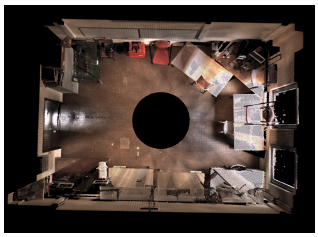
`Expt`	Envir -onment	# dense cloud	# sparse cloud	t_x (mm)	t_y (mm)	t_z (mm)	θ_x (°)	θ_y (°)	θ_z (°)	$RMSE$ (m)	# iter.
1	Office 1	411 924	69 952	150.0 155.0	170.0 172.9	35.0 34.8	5.0 4.7	0.0 0.4	0.0 0.1	- 0.0198	- 40
2	PAVIN 1	665 260	67 488	0.0 0.2	30.0 29.8	200.0 191.9	5.0 4.8	5.0 4.8	0.0 0.1	- 0.0188	- 36
3	Office 2	986 344	69 984	350.0 357.3	350.0 342.8	0.0 0.3	0.0 0.3	0.0 0.1	0.0 0.8	- 0.0199	- 39
4	PAVIN 2	1 364 245	67 768	65.0 64.1	45.0 45.7	200.0 200.9	0.0 0.3	7.0 6.9	5.0 4.9	- 0.0184	- 34
5	Office 3	2 550 564	69 728	20.0 21.7	110.0 111.1	70.0 66.9	10.0 10.1	5.0 4.2	0.0 0.1	- 0.0191	- 39
6	PAVIN 3	3 218 879	67 936	0.0 0.5	0.0 0.1	0.0 0.3	10.0 9.4	10.0 9.8	0.0 0.2	- 0.0184	- 55
7	Office 4	4 490 859	69 996	30.0 27.8	470.0 470.3	300.0 307.3	20.0 19.6	0.0 0.1	0.0 0.1	- 0.0190	- 57
8	PAVIN 4	5 025 457	67 904	50.0 52.1	50.0 48.2	50.0 53.3	5.0 5.3	5.0 7.5	5.0 5.3	- 0.0152	- 56
9	Office 5	7 076 192	69 760	50.0 55.2	50.0 50.6	50.0 53.9	5.0 5.5	5.0 4.2	5.0 4.6	- 0.0190	- 35
10	PAVIN 5	19 615 433	67 488	0.0 0.1	50.0 49.8	200.0 200.3	10.0 9.7	0.0 0.1	5.0 5.2	- 0.0169	- 48

evolution of the $RMSE$ as a function of the number of iterations (see Figure 8). As expected, the convergence error draws to a minimum until the imposed stopping condition is reached. It is normally a tolerance on the translation and rotation rates. In our case, this tolerance is 10^{-3} and 10^{-4} for the translation and rotation, respectively. We run the algorithm for several indoor and outdoor scenes, with different viewpoints, as depicted in Table 3 and shown in Figure 8. Each experiment is performed more than 20 times. For instance, for the experiment 1 (`Expt_1`), the displacement between the two clouds is $[150, 170, 35, 5, 0, 0]$, where the first three values correspond to the translation in millimeters and the last three to the rotation in degrees. As for the fourth experiment, the displacement is $[65, 45, 200, 0, 7, 5]$, which took 34 iterations for the algorithm to converge. A more complete analysis is summarized in Table 3, along with the $RMSE$ recorded and the number of iterations achieved at convergence. The analysis of this table allows us to identify three elements that influence the registration results. Namely, the inter-frame displacement and the difference in density between the two clouds, as well as the nature of the environment (indoor or outdoor). For the displacement, we can observe that the larger the initial displacement, the more difficult the registration is. We would like to draw the reader's attention to the fact that the displacements experimented here are quite large, keeping in mind that dense techniques generally require an inter-frame displacement since the cost function is linearized around $x = 0$.

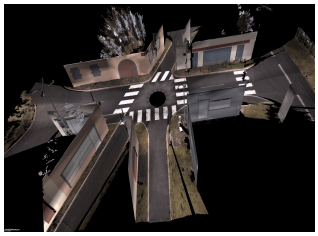
Continuing with our discussion on the influence of ini-



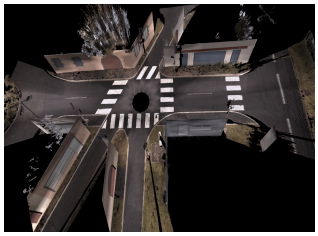
(a) Indoor point cloud delivered by the LiDAR Leica P20 with 88 539 380 points



(b) Indoor point cloud after sampling containing 986 344 points



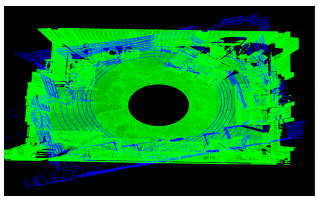
(c) Outdoor point cloud delivered by the LiDAR Leica P20 with 72 947 044 points



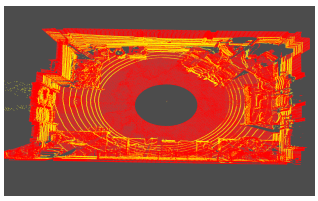
(d) Outdoor point cloud after sampling containing 2 732 783 points

Figure 6: Point cloud sampling.

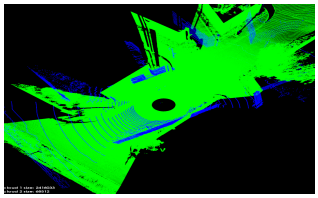
tial displacement, let us take the example of experiments `Expt_3` and `Expt_6`, which represent a pure translation and a pure rotation, respectively. These two experiments, as a sample of several experiments that we carry out, show that generally, the pure rotation requires more energy to reach the convergence with respect to the case of pure translation. Tables 10 to 15 demonstrate further rigorous testing of the cost function with our selection strategy. All six degrees of freedom (DoF) are activated and tested either independently or permuted arbitrarily. The results show the convergence values against the actual values, always with an initialization at identity, i.e. $x = [0, 0, 0, 0, 0, 0]$. It



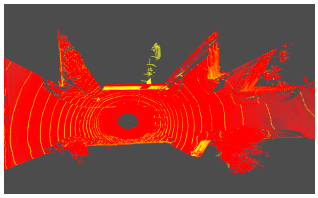
(a) Clouds before registration



(b) Clouds after registration



(c) Clouds before registration



(d) Clouds after registration

Figure 7: Registration results with CICP algorithm.

should be noted that, although the actual values are subjected to systematic errors of $\pm 2°$ in rotation and ± 1 cm in translation, they do not affect, one way or the other, the correct functioning in the various steps of our method. The true discrepancy between the two corresponding point clouds at convergence is measured using the $RMSE$. Regarding the influence of density, a quick look shows that the denser the clouds becomes, the more $RMSE$ decreases, leading to better registration. We will examine this parameter in detail in the Section 6.3. Finally, we observe that CICP performance depends on the scene. In fact, the impact of scene affects the performance of registration as observed by the difference of the number of iterations required to reach the convergence between PAVIN's and that of the office. It is clear that the indoor environment achieves better registration than the outdoor scene. This is possibly caused by the richness in planar regions of the former. It should not be overlooked that the outdoor environment contains a large amount of noise and outliers. This can be seen on `Expt_8` and `Expt_9`, in which the initial displacement is the same in both experiments. However, the alignment for the office dataset requires 34 iterations to converge instead of 56 for the PAVIN dataset.

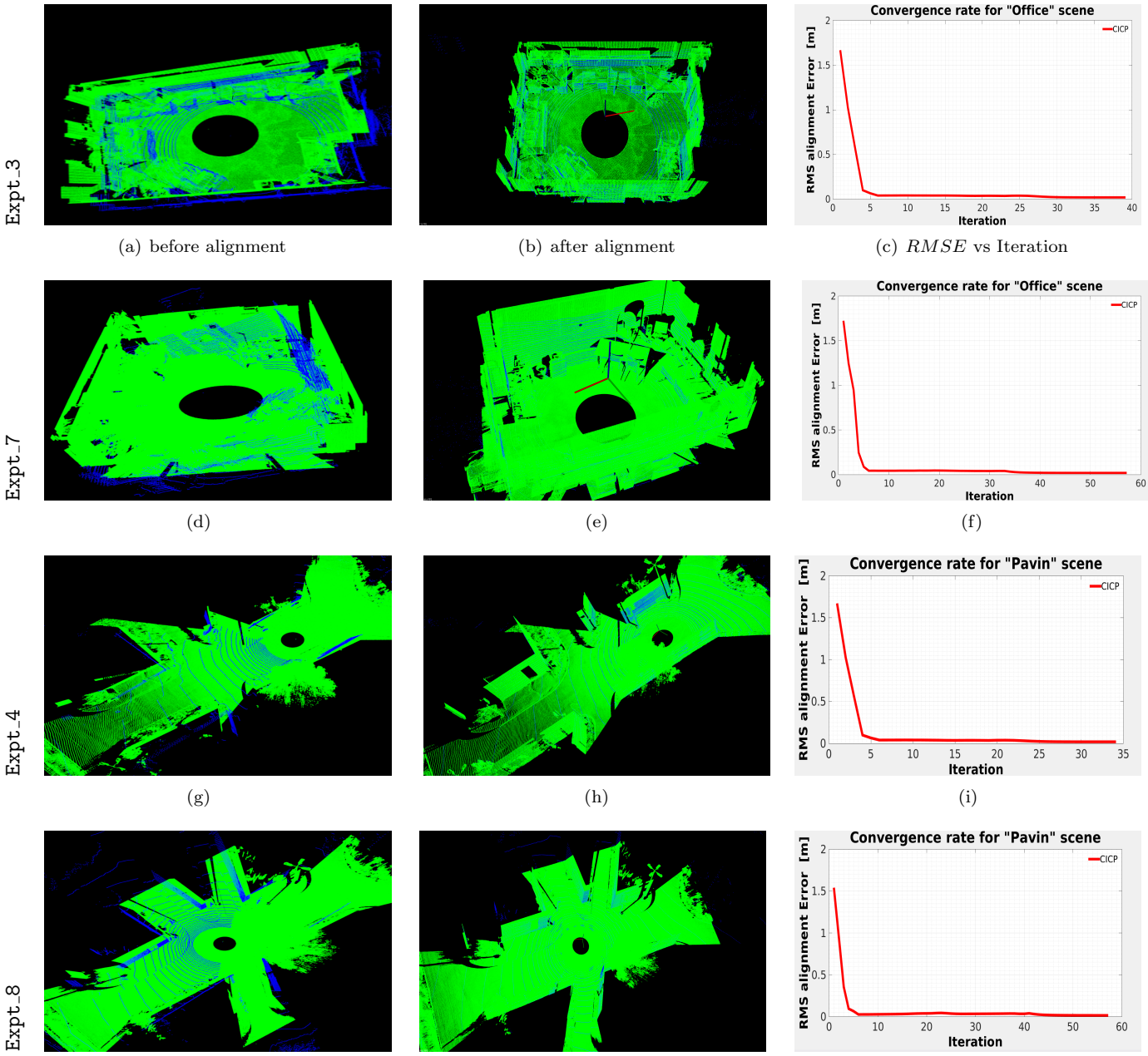
For the sake of illustration, we take four experiments arbitrarily (`Expt_3`, `Expt_4`, `Expt_7`, `Expt_8`), and show their state before and after the registration with their convergence profile in Figure 8. A closer look to the $RMSE$ curves in the third column of this figure shows that the residues which are far away are successfully minimized. However, the convergence begins very quickly and then stabilizes for a while before it reaches its minimum. This is mainly due to the fact that there is not a perfect point-to-point equivalence in the two pairing sets. This is quite logical and it can be explained by the large difference in density between the two clouds, the noise, and the clustering defects on the two clouds.

6.2. Comparison with Existing Methods

In order to compare our method with the existing state-of-the-art methods, we use implemented routines of PCL [26] library for the NDT algorithm, GICP, point-to-plane ICP and simple ICP for dense methods. For the case of sparse methods (methods based on features extraction) we also use PCL implementations of SIFT3D and FPFH to extract characteristic points from the two clouds, and use simple ICP to perform matching. The performance of each method is evaluated using three metrics: the accuracy, the relative translational error and the relative rotational error. The former describes the evolution of the root-mean-square point-to-point distance; this can be expressed mathematically as:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n} \parallel E_i \parallel^2} \tag{19}$$

where n is the number of points and E_i is the distance error between the source points and its correspondent in



(a) before alignment (b) after alignment (c) $RMSE$ vs Iteration

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 8: CICP results applied to different datasets from various environments.

the target cloud in each iteration. This can be expressed as follows:

$$E_i = \Sigma_{i=0}^{m} p_i - q_i \tag{20}$$

where m is the total number of points in the sparse cloud. p_i and q_i which represent two points of the source and target cloud, respectively, whereby p_i is transformed in the reference frame of q_i.

The second metric is the Relative Translational Error (RTE), which measures the translation gap between the ground truth ($\mathbf{t}_{GT}$) and the estimated ($\mathbf{t}_E$) translation vectors.

$$RTE = \|\mathbf{t}_{GT} - \mathbf{t}_E\|_2 \tag{21}$$

For the Relative Rotational Error (RRE), we use the metric defined on the tangent space of $\mathbb{SO}(3)$:

$$RRE = \|\log m(\mathbf{R}_E^T \mathbf{R}_{GT})\|_F \tag{22}$$

where logm(.) is the matrix logarithm, $\mathbf{R}_E$ is the estimated rotation matrix, $\mathbf{R}_{GT}$ is the ground truth rotation matrix and $\|.\|_F$ is the Frobenious norm.

Table 4 presents the results gathered in processing two indoor and outdoor scenes with the state-of-the-art meth-

Table 4: Comparison with the state-of-the-art methods.

		Dense					Sparse	
		ICP	pt2pl ICP	NDT	GICP	CICP	SIFT 3D+ICP	FPFH +ICP
Office	$RMSE$ (m)	0.0602	0.0620	0.0636	0.0636	**0.0299**	0.0516	failed
[m, m, m,° ,° ,°]	RTE (m)	0.2811	0.2482	0.2019	0.2178	**0.0169**	0.0191	failed
[0, 0.5, 0.5, 20, 0, 10]	RRE (°)	0.0517	0.0.0186	0.0285	0.0213	**0.0005**	0.0201	failed
PAVIN	$RMSE$ (m)	0.0836	0.0804	0.0824	0.0860	**0.0347**	0.0678	failed
[m, m, m,° ,° ,°]	RTE (m)	0.0365	0.0253	0.0315	0.0642	**0.0092**	0.021	failed
[0, 0.5, 0.3, 0, 0, 10]	RRE (°)	0.0058	0.0050	0.0058	0.0090	**0.0034**	0.0058	failed

ods. Bold values show the best result. Quantitatively, the $RMSE$ value of the indoor scene reaches 6 cm in the case of point-to-point, 6.2 cm point-to-plane ICP, 6.3 cm for the NDT and the GICP, more than 5 cm for SIFT3D and less than 3 cm for the proposed method. The maximum number of iterations for each test is fixed at 500 beyond which the algorithm is considered as not having converged if it reaches that ceiling, as is the case of the FPFH method.

Figure 9 shows the comparison of convergences between different registration methods. CICP outperforms the state-of-the-art methods on both datasets. In addition, it is shown that CICP is robust against scene variation.

6.2.1. Experiment on semi structured environment

The objective of this experimental set is to compare the performance of CICP with state-of-the-art algorithms (ICP, P2Pl, NDT and GICP) using a newly acquired dataset for a semi structured scenario type of environment where planar surfaces are far less pronounced. The results are given in Table 5, which presents the ground truth displacement versus the final results found by each algorithm. The tuning parameters using for each one of them is laid out in Table 6. Figure 10 gives the $RMSE$ error against the number of iterations to convergence. Again, the performance of CICP is very apparent and surpasses state-of-the-art, as shown in Figure 11.

Table 5: Performance comparison of each algorithm related to experimental section (§ 6.2.1).

	t_x (m)	t_y (m)	t_z (m)	θ_x (°)	θ_y (°)	θ_z (°)	$RMSE$ (m)	# iter
Actual	1.00	0.50	0.00	0.00	0.00	20.00	-	-
ICP	1.66	−0.28	0.00	−0.01	0.00	17.50	0.3406	500
P2Pl ICP	0.99	−0.52	0.00	−0.01	0.00	20.45	0.2482	361
NDT	1.71	−0.18	0.00	−0.01	−0.01	19.22	0.2876	131
GICP	1.14	0.16	−0.01	−0.06	0.00	15.02	0.5785	188
CICP	**0.99**	**0.50**	**0.00**	**0.02**	**0.05**	**20.24**	**0.0444**	**163**

Table 6: Tuning parameters used of each algorithm for experimental section (§ 6.2.1).

Parameter	ICP	ICP P2PL	NDT	GICP	CICP
$\varepsilon(stop\,criteria)$(m)	10^{-6}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
Rejection distance (m)	0.5	0.5	0.5	0.5	0.5
Maximum iterations	500	500	500	500	500
# neighbors normals estimation	-	10	-	-	10
# neighbors compute covariance	-	-	-	20	-
maximum step size for MT search	-	-	0.1	-	-
Resolution of NDT grid (m)	-	-	1	-	-

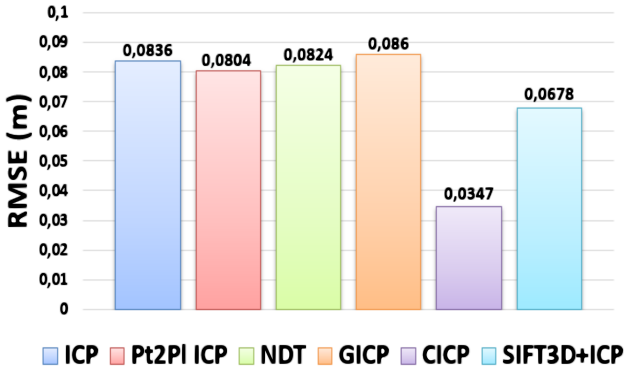
6.3. Changes in Density

To investigate the role of density on CICP performance, we conduct two experiments. The first is on two clouds collected with two different sensors, while for the second, the clouds are acquired with the same sensor but by varying the scanning resolution.

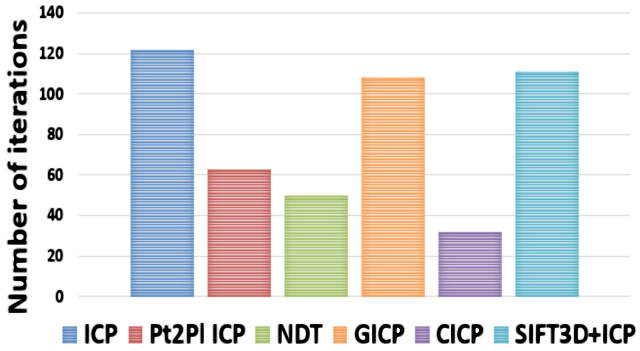
6.3.1. Data from two different sensors

Our first fold of experiments in this section is performed on clouds from two different sensors. The original dense cloud for the first dataset "office" is acquired with the LiDAR Leica P20. It is of size of 19 615 433 points. We perform different sampling using the method described in [72], which result in seven clouds having a size of between 4 million points and 100 000 points. These clouds are registered with a sparse cloud obtained from the Velodyne HDL-32E sensor of size 69 952 points. The results of this experiment are presented in Figure 12.

This experiment shows that despite the substantial difference in density between the two clouds in each test, there is only a slight change in the $RMSE$ (in the order of few millimeters) and in the iterations required to reach the convergence. This is mainly due to the fact that the density does not directly affect the error which is calculated from the pairing set of points. Density acts mainly on the correctness of the clustering. Indeed, the high density gives rise to properly grouped regions, which lead to points exhibiting the surface characteristics that it represents as much as possible, implying good matching and thus high accuracy. This represents the strength of our



(a) Comparison of $RMSE$ results of the different registration methods



(b) Comparison of number of iterations achieved at convergence of different registration methods

Figure 9: Convergence comparison between different registration methods.

method. Clustering always ensures the availability of representative points from which the matching is performed, even in the case of low density. The only difference is in the minimal change present in $RMSE$, as illustrated by the graphs in Figures 12a and 12c. This can be expressed as: high density implies a superior accuracy.

6.3.2. Data from the same sensor

The second batch of experimentation in this section consists in aligning different clouds acquired by the same sensor. Figure 13 shows seven clouds with different densities. All these clouds are taken with the same sensor by changing the scanning resolution each time. Indeed, the LiDAR Leica P20 allows the change of resolution, by increasing or decreasing the sampling distance (distance that separates two points of the cloud at a given distance from the scanner). In the case of the Leica P20, this distance varies from 0.8 mm per 10 m to 50 mm/ 10 m, giving rise to different density variation as illustrated in Figure 13. In this experiment, the dimensions of clouds are fixed within the sensor. This latter is placed in one fixed position, and the only difference between these 7 clouds is the scanning resolution (neither the dimensions, nor the viewpoint). The outcome of this experiment is shown in Table 7.

The conclusion that can be drawn from the findings of this evaluation is that the difference of density between the two clouds only affects the convergence of the algorithm. As in this experiment, nothing changes between the two clouds except the density. The final $RMSE$ values are all equal and are close to the accuracy of the sensor (3 mm). The only difference is in the number of iterations needed to reach the convergence. This can be formulated by: a small difference in density between the two clouds extends the convergence process. In fact, this is due to several reasons. First, choosing the sparse cloud as the source cloud (in order to optimize the computing time as mentioned above). This means that the search for nearest neighbors is done in the dense cloud for the points of sparse cloud. Secondly, according to the mathematical definition proposed in this article (Section 4.1), the voxel size for clustering is fixed according to the number of points in the sparse cloud, in order to ensure points for matching. These two reasons impact the search of the nearest neighbor, which is easier and more accurate in a dense cloud and less precise and longer in a less dense cloud.

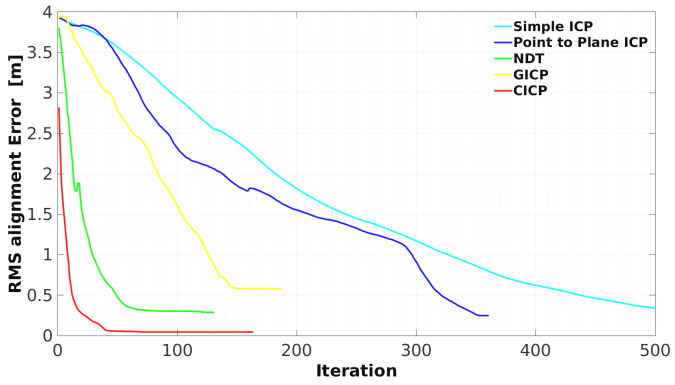


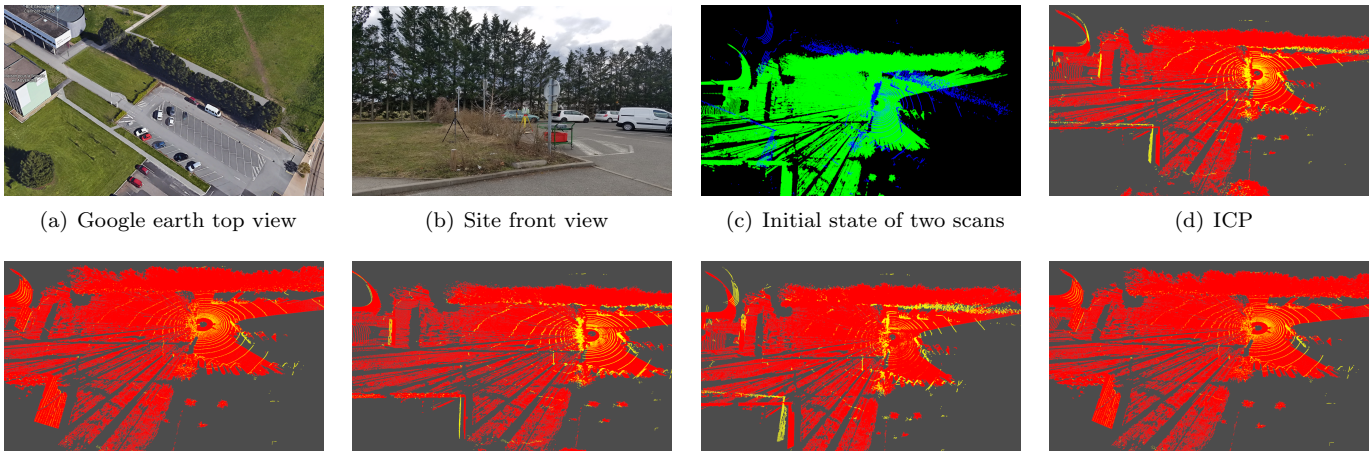
Figure 10: Cost function evolution for the unstructured environment.

6.4. Changes in density and viewpoint

Table 8 shows the results of the alignment of different clouds acquired by the same sensor and with viewpoint

Table 7: Density change results.

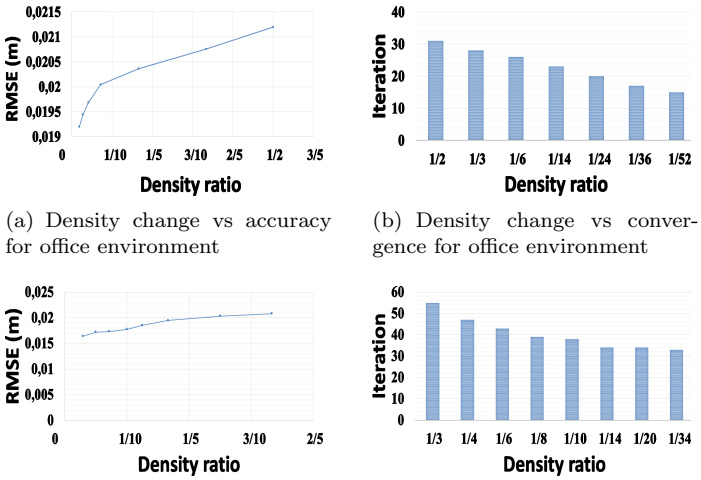
Resolution cloud 1 (mm at 10 m)	1.6	3.1	6.3	12.5
Resolution could 2 (mm at 10 m)	25	25	25	25
Resolution ratio	1/16	1/8	1/4	1/2
$RMSE$ (m)	0.0027	0.0027	0.0027	0.0027
Iteration	3	6	7	10



(a) Google earth top view (b) Site front view (c) Initial state of two scans (d) ICP

(e) Pt2pl ICP (f) NDT (g) GICP (h) CICP

Figure 11: Results with a semi structured environment. Registration between a Leica P20 scan and the Velodyne HDL32-E is performed using state-of-the-art algorithms and CICP.



(a) Density change vs accuracy for office environment

(b) Density change vs convergence for office environment

(c) Density change vs accuracy for PAVIN environment

(d) Density change vs convergence for PAVIN environment

Figure 12: Density change effects on the convergence and accuracy of the CICP method.

change, which was $[50, 50, 50, 5, 0, 0]$. Our main motivation to conduct this experiment, which differs from the previous one by the addition of the view change, is to confirm the influence of the density change as the scanned pattern is the same.

Findings from this second experiment provide further evidence about the conclusions drawn earlier and the role of density. Indeed, as we explained previously, high density gives rise to superior accuracy and improves the convergence speed (this can be seen on the sixth and seventh rows of the Table 8).

6.5. Comparison with Various Sensors

To investigate the potential of our approach to align point clouds from different sensors, we test it with three

Table 8: Density & viewpoint change results.

Resolution cloud 1 (mm at 10 m)	0.8	1.6	3.1	6.3	12.5	25
# points of cloud1	23 870 726	9 800 513	2 527 043	1 219 973	673 537	65 626
Resolution cloud 2 (mm at 10 m)	50	50	50	50	50	50
# points of cloud 2	20 676	20 676	20 676	20 676	20 676	20 676
Ratio of points	1/1150	1/470	1/120	1/60	1/32	1/3
$RMSE$ (m)	0.0060	0.0065	0.0073	0.0082	0.0107	0.0166
# iteration	40	44	46	50	58	64

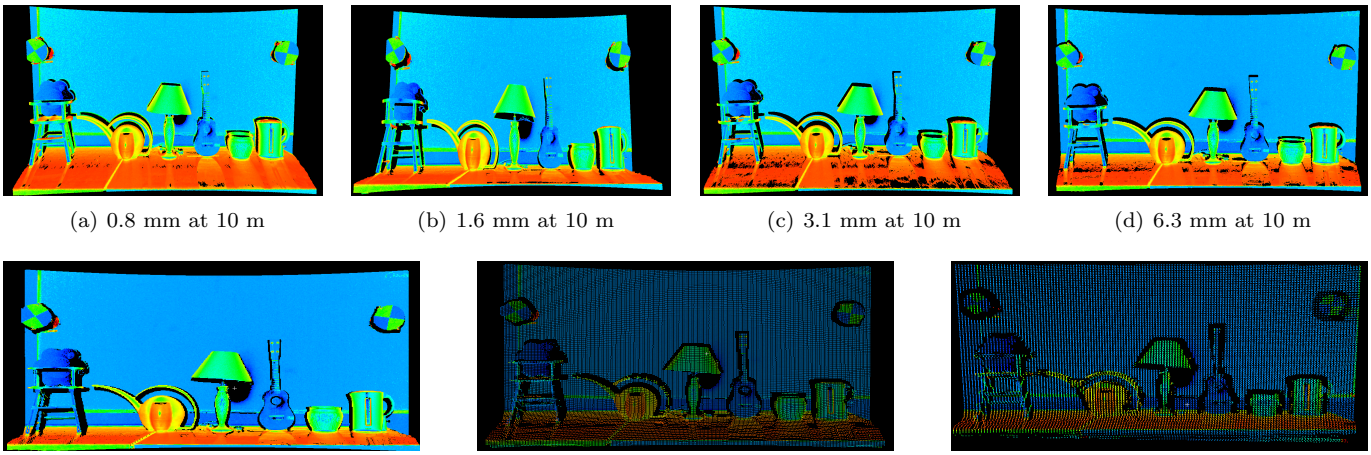
kinds of sensors, the Leica P20 LiDAR, Velodyne HDL32-E LiDAR and SR4000 Time-of-Flight camera. Figure 14 shows these different sensors and Table 9 exhibits their hardware specifications.

6.5.1. ***Leica P20***

Many varieties of 3D LiDAR sensors are available on the market, but they all work with the same basic principle [79]. They emit pulses and detect their reflection in order to explore the object or the environment. Leica P20 is a Time-of-Flight scanner which offers greater range and precision.

6.5.2. ***Velodyne HDL32-E***

The Velodyne HDL-32 produces 3D scans by rotating a 32-beam array around its vertical axis at 10 Hz. It produces approximately 700 000 points per second or 2200 points per laser beam at a range of 1 through 70 meters. This sensor provides an angular resolution of approximately 0.16° with a field of view (FOV) of 360°. Its vertical field of view is from −30.67° to +10.67° with an



(a) 0.8 mm at 10 m (b) 1.6 mm at 10 m (c) 3.1 mm at 10 m (d) 6.3 mm at 10 m

(e) 12.5 mm at 10 m (f) 25 mm at 10 m (g) 50 mm at 10 m

Figure 13: Density change within the same sensor (the colors from blue to red correspond to the scale of the intensity of the reflectance).

Figure 14: Sensors used to test the CICP approach. From left to right: SR4000 Time-of-Flight camera, Velodyne HDL 32-E, and Leica P20 Laser.

Table 9: Sensors hardware specifications.

Sensor	Range (m)	Field of view (°) Horizontal	Vertical	Scanning Frequency (Hz)	Accuracy (mm)
Leica P20	$[0.5 \dots 120]$	360	270	50	3
Velodyne HDL-32E	$[1 \dots 70]$	360	$[-30 \pm 10]$	10	20
SR4000	$[0.1 \dots 10]$	43.6	34.6	50	15

angular resolution of 1.33°. Its measurement accuracy is generally less than 2 cm.

6.5.3. ***SR4000 Time of Flight camera***

The Time of Flight (ToF) camera is a two-dimensional scanner which captures full depth per frame and with a single light pulse.

The rest of this section discusses the registration of different clouds acquired by these sensors.

6.5.4. Leica P20 vs Velodyne

In this experiment, the point cloud produced by the Velodyne sensor is the sparse cloud and the second cloud, which represents the dense one, is produced by the LiDAR Leica P20 (Figure 15). The algorithm converges after 64 iterations, with 0.0199 mm as the $RMSE$ value.

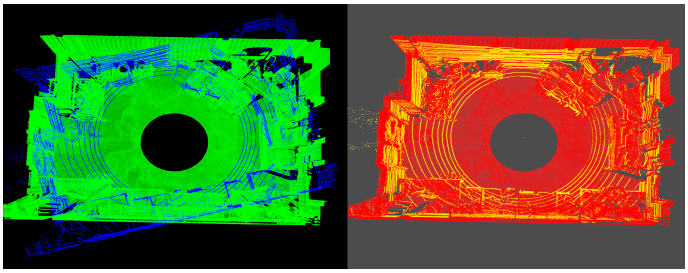


Figure 15: Registration of two clouds captured by Leica P20 LiDAR for the dense cloud and Velodyne HDL-32E for the sparser cloud.

6.5.5. Leica P20 vs SR4000

Here, the sparse cloud is produced by the SR4000 Time-of-Flight camera, while the dense cloud is produced by the Leica P20 sensor (Figure 16). The latter is less affected by noise than the former.

The results of this convergence are 100 and 0.0203 for the number of iterations and the $RMSE$, respectively. The reader can observe that the $RMSE$ of P20 vs SR4000 experiment is higher than the $RMSE$ of the test performed between P20 and Velodyne sensors. This is justified by the large presence of noise in the cloud delivered by the ToF camera.

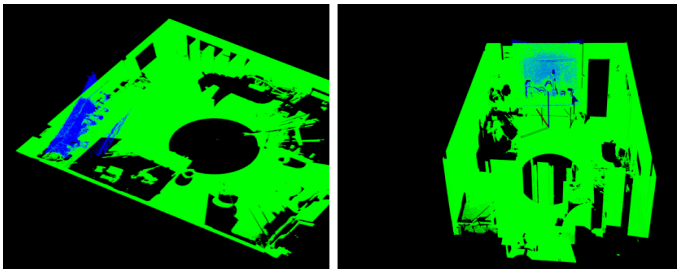


Figure 16: Registration of two clouds captured by Leica P20 LiDAR for the dense cloud and SR4000 ToF camera for the sparser cloud.

6.5.6. SR4000 vs Velodyne

Here the two sensors are affected by noise. This explains why the registration takes more than 124 iterations to converge. The final $RMSE$ is about 0.0231. In contrast to the previous experiment, the dense cloud is produced by the ToF camera, while the Velodyne cloud represents the sparse one. Even though the total number of points in the Velodyne cloud is almost 3 times higher than the cloud ToF, in the part that interests us (representing approximately $(2 \times 1 \times 1)$ m^3, which enclose the table and objects exposed on it and the table behind), the ToF cloud is denser than the cloud of the Velodyne (Figure 17).

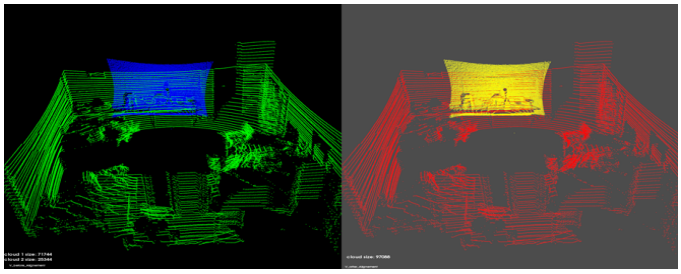


Figure 17: Registration of two clouds captured by a ToF camera for the dense cloud and a Velodyne for the sparse cloud.

6.6. Demonstration with Dense-to-Dense Data

At the end of this experimental section, we wish to highlight that the CICP method can be used with clouds of the same nature (dense to dense or sparse to sparse).

Figure 18 shows the state of the two dense clouds before and after the registration. Despite the large number of points, the final result is correctly aligned. Here is another benefit of our approach, the fact of not considering the entire set of points for matching, but only a collected set of points from each local surface, which improves the convergence speed.

6.7. Impact of the voxel size

The voxel size plays a very important role in the convergence of the algorithm. Figure 19 illustrates how the convergence is impacted by the change of this parameter. When increasing the voxel size, the number of points included in this voxel is increased, which reduces the number of points used for matching. As clustering generates a few representative points relative to the input points, thereby increasing the convergence speed, but decreasing the accuracy. Setting a small voxel size decreases the convergence speed, but increases accuracy, due to the availability of sufficient points for matching. Therefore, a reliable trade off needs to be determined in order to find the optimal discretization of the point cloud.

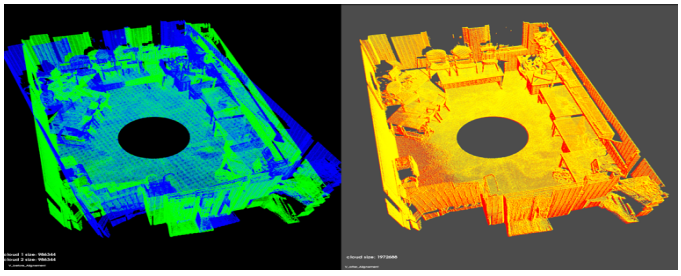


Figure 18: Registration of two dense clouds of an indoor scene captured by a Leica P20 sensor.

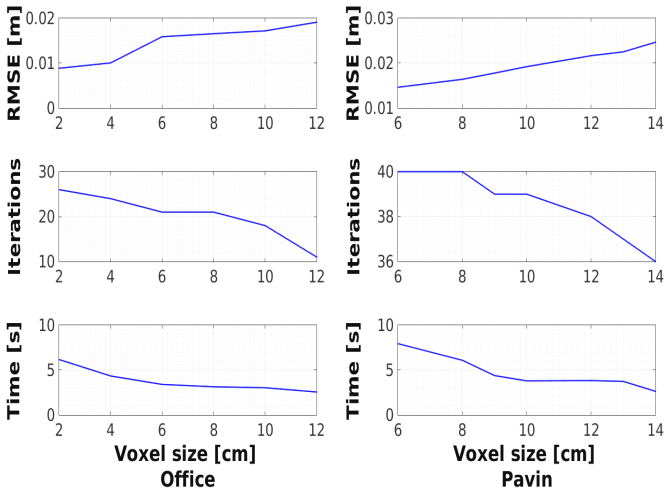


Figure 19: Clustering voxel size effect on registration accuracy and convergence speed.

7. Discussion

1. Two or more point clouds are acquired from the same scene but with different sensors (e.g. vision based system producing sparse cloud and 3D LiDAR producing dense cloud), leading to different clouds with their own local coordinate system, resolution and number of points. Registration methods based on the classical point-to-point ICP metrics fail to provide an accurate pose estimate because of the large discrepancies in density between the two point clouds. The difficulty lies in the fact that there are no direct correspondences between the source and the target point clouds. What is more, methods based on the geometric characteristics are also unsuitable as these (mainly normal and curvature [70]), which are essentially based on their estimations on the neighboring points, are affected by the change in resolution and scanning patterns [45, 13].
2. The information carried by a 3D point can contain color, reflectance (intensity), positions, normals and curvatures. Whilst color or reflectance differs from one sensor to another, for the case of a static environment, the global geometric aspect of the scene remains unchanged. This is obviously, why CICP capitalizes on geometric primitives. In addition, this makes it independent of weather and illumination conditions.

3. It complements outperforms the famous the state-of-the-art methods (ICP, NDT, GICP) for this kind of multi sensors applications. These classical algorithms find their limits with dense-sparse registration, because they are all based on point-to-point matching. Whereas our method is based on the concept of surface-to-surface matching, this concept can be generalized to any type of common clustering between the two point clouds. We can imagine the use of segments or dominant direction of variation of the points which corresponds to the highest eigenvalue in the PCA. The matched surfaces are chosen to be very small local surfaces. To do that, a common voxelization between the two clouds with a very small voxel size is performed. The choice of matching small surfaces was made in order to preserve the topological details of the scanned environment and to guarantee a considerable number of matched points between the two clouds.
4. Normals are computed once before starting the process and are used only to distinguish the different local surfaces. They are not used in the alignment process. The use of surface normals in point to plane ICP and its variants is motivated by the fact that the former are robustly estimated in the presence of noisy surfaces. Otherwise, they will cause ICP to diverge. In the case of CICP, we make use of normals to perform surface segmentation. Such an approach improves the surface estimate for noisy measurements. Figure 2 shows normal vectors extracted from the same surface scanned by two distinct sensors. It can be clearly observed that though the surfaces are piecewise planar, their estimated normals do not correspond. Therefore, this reinforces the idea of not retaining the normals for the optimization phase.
5. Whether in the case of dense or sparse cloud, there is a certain number of points that are redundant. Redundancy, though useful to make robust the overdetermined system of the normal equation comes at the cost of increased computation. Therefore, the use of representative subsets of points give the same if not better accuracy with less computational time.
6. As shown in this paper, the use of normals is a double-edged sword and it can guarantee good quality results if accurately exploited. In the same way, normals can amplify noise leading to divergence of the ICP method. The proposed approach makes use of normals for clustering points from the same surface, and we avoid using them to establish correspondences due to various disturbances coming from the sensor. This strategy enables us to overcome the main weakness of dense to sparse/sparse to dense registration. It allows us to surpass the problem of density in search of correspondences. The cascaded effects of an improved surface match correspondence lead to better accuracy in the registration pipeline at reduced computational cost.

8. Conclusion

In this paper, a novel approach for sparse to dense point cloud registration has been presented. The traditional ICP pipeline is modified to accommodate a smarter way of surface patch correspondence consisting of three main blocks; voxelization, clustering, representative election. The motivation behind this strategy is to match identical surfaces in the 3D world but scanned using different type of depth sensors. With various sensors come different resolutions and hence different point cloud representations. Throughout our experimental phase, we demonstrate the efficiency of our algorithm in terms of alignment accuracy where other state-of-the-art techniques perform poorly since they do not cater for these above-mentioned differences. Furthermore, we show that the alignment technique works perfectly even by changing the density of the points of the two clouds.

To summarize, our proposed methodology provides the following improvements:

- patch surface segmentation contributing to noise reduction,
- improved selection by a novel surface point representative approach,
- reduced amount of processed data during matching phase,
- dense to sparse registration applicable to the various depth sensors on the market,
- a novel mathematical definition of sparse and dense clouds.

Our algorithm has been successfully tested on various indoor an outdoor datasets. Our future work will be concentrated on the notion of uncertainty in the model to probabilistically incorporate a fusion stage in our pipeline. Furthermore, on top of normal feature extraction, other primitives might also be considered such as surface patch curvatures or their relative intensity (if available) to be able to perform a robust check on the matching and correspondence phase. Finally, the scope of this paper lies within the generic problem of localization for either handheld applications for augmented or virtual reality or robotic platforms navigating inside an *a priori* mapped environment. The latter is among potential applications of our approach, since CICP is suitable to localize a vehicle equipped with a sensor that provides sparse data (e.g. Velodyne) in a dense and accurate map. In this perspective, an effort towards code optimization will therefore be considered.

Disclosure statement

"The authors declare no conflict of interest."

Table 10: Variation only of rotation around one axis: θ_z. The second column represents the actual registration parameters and the third one represents the CICP estimated values.

θ_z (°)	Actual	Final state	Iterations	$RMSE$
5	$[0, 0, 0, 0, 0, 5]$	[0.00 , 0.01 , 0.00 , 0.03 , 0.05 , 4.73]	64	0.0197
10	$[0, 0, 0, 0, 0, 10]$	[0.00 , 0.01 , 0.00 , 0.01 , 0.15 , 10.01]	99	0.0196
15	$[0, 0, 0, 0, 0, 15]$	[0.00 , 0.00 , 0.00 , 0.62 , 0.13 , 14.68]	95	0.0198
20	$[0, 0, 0, 0, 0, 20]$	[0.00 , 0.01 , 0.01 , 0.65 , 0.10 , 19.71]	157	0.0197
25	$[0, 0, 0, 0, 0, 25]$	[0.00 , 0.00 , 0.00 , 0.79 , 0.25 , 24.64]	220	0.0196
30	$[0, 0, 0, 0, 0, 30]$	[0.00 , 0.00 , 0.03 , 1.85 , 0.12 , 29.59]	303	0.022
35	$[0, 0, 0, 0, 0, 35]$	[0.05 , 0.03 , 0.01 , 0.95 , 0.14 , 36.05]	418	0.0221
40	$[0, 0, 0, 0, 0, 40]$	[0.06 , 0.03 , 0.01 , 0.85 , 0.10 , 40.64]	547	0.0215

Table 11: Variation only of rotation around two axes.

θ_y(°)	θ_z (°)	Initial	Final state	Iteration	$RMSE$
15	10	$[0, 0, 0, 0, 15, 10]$	[0.00 , 0.00 , −0.01 , 0.09 , 14.59 , 9.62]	83	0.0183
15	15	$[0, 0, 0, 0, 15, 15]$	[0.00 , 0.00 , −0.01 , 0.50 , 15.03 , 14.71]	140	0.0188
15	25	$[0, 0, 0, 0, 15, 25]$	[0.00 , 0.00 , 0.00 , 0.65 , 15.00 , 24.92]	387	0.0184
20	15	$[0, 0, 0, 0, 20, 15]$	[0.00 , 0.00 , 0.01 , 0.89 , 19.49 , 14.91]	167	0.0184
20	25	$[0, 0, 0, 0, 20, 25]$	[0.00 , 0.00 , −0.01 , 0.82 , 19.61 , 24.68]	337	0.0179
30	20	$[0, 0, 0, 0, 30, 20]$	[0.01 , 0.00 , 0.02 , 0.93 , 28.81 , 19.30]	394	0.01807
30	25	$[0, 0, 0, 0, 30, 25]$	[0.01 , 0.00 , 0.03 , 0.91 , 28.58 , 24.68]	616	0.01807

Table 12: Variation only of the translation along one axis: t_y.

t_z (cm)	Actual	Final state	Iteration	$RMSE$
10	[0, 0.10 , 0, 0, 0, 0]	[0.00 , 0.10 , 0.00 , 0.23 , −0.33 , −0.08]	25	0.0204
20	[0, 0.20, 0, 0, 0, 0]	[0.00 , 0.20 , 0.00 , 0.30 , 0.04 , −0.42]	27	0.0205
30	[0, 0.30 , 0, 0, 0, 0]	[0.00 , 0.30 , 0.00 , 0.02 , 0.27 , 0.78]	35	0.0202
40	[0, 0.40 , 0, 0, 0, 0]	[0.00 , 0.41 , 0.00 , 0.18 , 0.26 , 0.77]	41	0.0205
50	[0, 0.50 , 0, 0, 0, 0]	[0.00 , 0.51 , 0.01 , 0.37 , 0.42 , 0.90]	65	0.0205
60	[0, 0.60 , 0, 0, 0, 0]	[0.00 , 0.61 , 0.01 , 0.51 , 0.41 , −0.59]	94	0.0228
70	[0, 0.70 , 0, 0, 0, 0]	[0.00 , 0.71 , 0.01 , 0.37 , 0.39 , −0.01]	117	0.0217
80	[0, 0.80 , 0, 0, 0, 0]	[0.00 , 0.80 , 0.01 , 0.19 , 0.48 , −0.32]	145	0.0211
90	[0, 0.90 , 0, 0, 0, 0]	[0.00 , 0.91 , 0.01 , 0.09 , 0.46 , 0.75]	160	0.0213
100	[0, 1.00 , 0, 0, 0, 0]	[0.00 , 1.01 , 0.00 , 0.09 , 0.30 , 0.05]	215	0.0212
120	[0, 1.20 , 0, 0, 0, 0]	[0.00 , 1.20 , 0.00 , 0.04 , −0.04 , 0.89]	278	0.0198
150	[0, 1.50 , 0, 0, 0, 0]	[0.00 , 1.51 , 0.00 , 0.20 , 0.33 , 0.76]	386	0.0209
170	[0, 1.70 , 0, 0, 0, 3]	[0.00 , 1.70 , 0.01 , 0.11 , 0.22 , 0.16]	477	0.0196
200	[0, 2.00 , 0, 0, 0, 4]	[−0.01 , 2.00 , 0.01 , 0.26 , 0.09 , 0.83]	618	0.0201

References

[1] G. Agamennoni, S. Fontana, R. Y. Siegwart, D. G. Sorrenti, Point Clouds Registration with Probabilistic Data Association, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4092–4098.

[2] B. Bellekens, V. Spruyt, R. B. Maarten Weyn, A survey of rigid 3D pointcloud registration algorithms, in: Fourth International

Table 13: Variation only of the translation along two axes: t_x and t_y.

$t_{x\,(\mathrm{cm})}$	$t_{y\ \ (\mathrm{cm})}$	Actual	Final state	Iteration	$RMSE$
30	30	[0.30 , 0.30 , 0, 0, 0, 0]	[0.30 , 0.31 , 0.00 , 0.17 , 0.46 , 0.36]	33	0.0186
30	50	[0.30 , 0.50 , 0, 0, 0, 0]	[0.31 , 0.51 , 0.00 , 0.53 , 0.48 , 0.41]	67	0.0198
30	70	[0.30 , 0.70 , 0, 0, 0, 0]	[0.31 , 0.71 , 0.01 , 0.54 , 0.49 , 0.41]	92	0.0198
30	100	[0.30 , 1.00 , 0, 0, 0, 0]	[0.32 , 1.01 , 0.02 , 0.55 , 0.49 , 0.41]	166	0.0198
50	50	[0.50 , 0.50 , 0, 0, 0, 0]	[0.50 , 0.51 , 0.01 , 0.35 , 0.53 , 0.55]	87	0.0204
50	70	[0.50 , 0.70 , 0, 0, 0, 0]	[0.50 , 0.71 , 0.00 , 0.36 , 0.53 , 0.55]	108	0.0204
50	100	[0.50 , 1.00 , 0, 0, 0, 0]	[0.51 , 1.01 , 0.01 , 0.37 , 0.54 , 0.55]	185	0.0205
50	150	[0.50 , 1.50 , 0, 0, 0, 0]	[0.51 , 1.51 , 0.02 , 0.39 , 0.54 , 0.55]	298	0.0204
100	100	[1.00 , 1.00 , 0, 0, 0, 0]	[0.99 , 0.99 , 0.01 , 0.66 , 0.46 , 0.26]	262	0.0199
100	120	[1.00 , 1.20 , 0, 0, 0, 0]	[1.00 , 1.19 , 0.01 , 0.72 , 0.50 , 0.26]	309	0.0200
100	150	[1.00 , 1.50 , 0, 0, 0, 0]	[1.00 , 1.49 , 0.00 , 0.72 , 0.50 , 0.26]	450	0.0199
100	200	[1.00 , 2.00 , 0, 0, 0, 0]	[1.00 , 1.99 , 0.00 , 0.72 , 0.50 , 0.26]	590	0.0200

Table 14: Variation only of the translation along three axes: t_x, t_y and t_z.

$t_{x\,(\mathrm{cm})}$	$t_{y\,(\mathrm{cm})}$	$t_{z\,(\mathrm{cm})}$	Actual	Final state	Iteration	$RMSE$
30	30	30	[0.30 , 0.30 , 0.30 , 0, 0, 0]	[0.31 , 0.31 , 0.29 , 0.50 , 0.47 , 0.42]	74	0.0197
50	50	50	[0.50 , 0.50 , 0.50 , 0, 0, 0]	[0.50 , 0.52 , 0.50 , 0.37 , 0.47 , 0.55]	124	0.0205
100	100	100	[1.00 , 1.00 , 1.00 , 0, 0, 0]	[1.00 , 1.01 , 1.00 , 0.64 , 0.44 , 0.27]	408	0.0200

Table 15: Variation of translation along one axis (t_y) and rotation around one axis (θ_z).

$t_{y\,(\mathrm{cm})}$	$\theta_{z\,(^\circ)}$	Actual	Final state	Iteration	$RMSE$
30	20	[0.00 , 0.30 , 0, 0, 0, 0.20]	[0.00 , 0.30 , 0.00 , 0.77 , 0.07 , 19.50]	85	0.0199
30	30	[0.00 , 0.30 , 0, 0, 0, 0.30]	[0.00 , 0.29 , 0.00 , 0.46 , 0.16 , 29.30]	246	0.0200
50	20	[0.00 , 0.50 , 0, 0, 0, 0.20]	[0.00 , 0.50 , 0.01 , 0.05 , 0.08 , 20.12]	98	0.0206
50	30	[0.00 , 0.50 , 0, 0, 0, 0.30]	[0.00 , 0.50 , 0.00 , 0.48 , 0.21 , 29.92]	350	0.0203
100	20	[0.00 , 1.00 , 0, 0, 0, 0.20]	[0.00 , 1.01 , 0.01 , 0.74 , 0.10 , 20.85]	186	0.0206
100	25	[0.00 , 1.00 , 0, 0, 0, 0.25]	[0.00 , 1.01 , 0.01 , 0.67 , 0.16 , 25.89]	227	0.0197
150	20	[0.00 , 1.50 , 0, 0, 0, 0.20]	[0.00 , 1.50 , 0.01 , 0.87 , 0.17 , 21.09]	473	0.0204

Conference on Ambient Computing, Applications, Services and Technologies, Proceedings, IARA, 2014, pp. 8–13.

[3] F. Pomerleau, F. Colas, R. Siegwart, A Review of Point Cloud Registration Algorithms for Mobile Robotics, Foundations and Trends in Robotics 4 (1-104) (2015) 1–104.

[4] M. Velas, M. Spanel, A. Herout, Collar Line Segments for fast odometry estimation from Velodyne point clouds, in: IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 4486–4495.

[5] T. Zhang, Surface Reconstruction with Sparse Point Clouds of Velodyne Sensor, in: The 14th IFToMM World Congress, 2015.

[6] J. Razlaw, D. Droeschel, D. Holz, S. Behnke, Evaluation of registration methods for sparse 3d laser scans, in: Mobile Robots (ECMR), 2015 European Conference on, IEEE, 2015, pp. 1–7.

[7] W. S. Grant, R. C. Voorhies, L. Itti, Finding planes in LiDAR point clouds for real-time registration, in: IEEE International Conference on Intelligent Robots and Systems, 2013, pp. 4347–4354.

[8] A. Patidar, P. Sanjiv, A Review Paper on Self - Driving Car ' s and its Applications, in: National Conference on Innovations in Micro-electronics, Signal Processing and Communication Technologies IJIRST, 2016, pp. 33–35.

[9] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, P. Mahoney, Stanley: The Robot That Won the DARPA Grand Challenge, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, Ch. 1, pp. 1–43. doi:10.1007/978-3-540-73429-1_1.
URL http://dx.doi.org/10.1007/978-3-540-73429-1_1

[10] R. W. Wolcott, R. M. Eustice, Visual localization within LIDAR maps for automated urban driving, in: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, IEEE, 2014, pp. 176–183.
[11] T. Caselitz, B. Steder, M. Ruhnke, W. Burgard, Monocular Camera Localization in 3D LiDAR Maps, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 1–6.
[12] W. Maddern, P. Newman, Real-time probabilistic fusion of sparse 3d lidar and dense stereo, in: Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, IEEE, 2016, pp. 2181–2188.
[13] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, S. Behnke, Registration with the Point Cloud Library PCL, IEEE Robotics & Automation Magazine 22 (4) (2015) 1–13.
[14] Y. Chen, G. Medioni, Object modeling by registration of multiple range images, in: IEEE International Conference on Robotics and Automation, 1991, pp. 2724–2729.
[15] P. J. Besl, N. D. McKay, A Method for Registration of 3-D Shapes, IEEE Trans. Pattern Anal. Mach. Intell. 14 (2) (1992) 239–256.
[16] P. Markelj, D. Tomaževič, B. Likar, F. Pernuš, A review of 3D/2D registration methods for image-guided interventions, Medical image analysis 16 (3) (2012) 642–661.
[17] J. Salvi, C. Matabosch, D. Fofi, J. Forest, A review of recent range image registration methods with accuracy evaluation, Image and Vision computing 25 (5) (2007) 578–596.
[18] F. Pomerleau, Applied registration for robotics, Ph.D. thesis, ETH ZURICH (2013).
[19] R. Fabio, From point cloud to surface: the modeling and visualization problem, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 34 (2003) 11.
[20] R. Marani, V. Reno, M. Nitti, T. D'Orazio, E. Stella, A modified iterative closest point algorithm for 3D point cloud registration, Computer-Aided Civil and Infrastructure Engineering 31 (7) (2016) 515–534.
[21] J. Serafin, E. Olson, G. Grisetti, Fast and Robust 3D Feature Extraction from Sparse Point Clouds, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4105–4112.
[22] C. M. Costa, H. M. Sobreira, A. J. Sousa, G. M. Veiga, Robust 3/6 DoF self-localization system with selective map update for mobile robot platforms, Robotics and Autonomous Systems 76 (C) (2016) 113–140.
[23] Y. Feng, A. Schlichting, C. Brenner, 3D feature point extraction from LiDAR data using a neural network, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives 41 (2016) 41 (2016) 563–569.
[24] S. Filipe, L. A. Alexandre, A comparative evaluation of 3D keypoint detectors in a RGB-D object dataset, in: Computer Vision Theory and Applications (VISAPP), 2014 International Conference on, Vol. 1, IEEE, 2014, pp. 476–483.
[25] D. G. Lowe, Object recognition from local scale-invariant features, in: Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, pp. 1150–1157.
[26] R. B. Rusu, S. Cousins, 3D is here: point cloud library, in: IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 1 – 4.
[27] R. Hänsch, T. Weber, O. Hellwich, Comparison of 3D interest point detectors and descriptors for point cloud fusion, ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences II-3 (September) (2014) 57–64.
[28] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, Computer Vision–ECCV 2006 (2006) 430–443.
[29] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (surf), Computer vision and image understanding 110 (3) (2008) 346–359.
[30] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, Orb: An efficient alternative to sift or surf, in: Computer Vision (ICCV), 2011 IEEE international conference on, IEEE, 2011, pp. 2564–2571.
[31] B. Steder, R. B. Rusu, K. Konolige, W. Burgard, Point Feature Extraction on 3D Range Scans Taking into Account Object Boundaries Bastian, in: IEEE International Conference on Robotics and Automation (ICRA),, 2011, pp. 2601–2608.
[32] R. B. Rusu, N. Blodow, Z. C. Marton, M. Beetz, Aligning Point Cloud Views using Persistent Feature Histograms, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2008, pp. 3384–3391.
[33] R. B. Rusu, N. Blodow, M. Beetz, Fast Point Feature Histograms (FPFH) for 3D registration, in: IEEE International Conference on Robotics and Automation, 2009, pp. 3212–3217.
[34] R. B. Rusu, G. Bradski, R. Thibaux, J. Hsu, Fast 3d recognition and pose using the viewpoint feature histogram, in: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, IEEE, 2010, pp. 2155–2162.
[35] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R. B. Rusu, G. Bradski, Cad-model recognition and 6dof pose estimation using 3d cues, in: Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, IEEE, 2011, pp. 585–592.
[36] I. T. Jolliffe, Principal Component Analysis for Special Types of Data, Springer New York, New York, NY, 1986, Ch. 11, pp. 199–222.
[37] Y. Zhong, Intrinsic shape signatures: A shape descriptor for 3d object recognition, in: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, IEEE, 2009, pp. 689–696.
[38] A. Mian, M. Bennamoun, R. Owens, On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes, International Journal of Computer Vision 89 (2) (2010) 348–361.
[39] T. Weber, R. Hänsch, O. Hellwich, Automatic registration of unordered point clouds acquired by Kinect sensors using an overlap heuristic, ISPRS Journal of Photogrammetry and Remote Sensing 102 (2015) 96–109.
[40] J. Serafin, G. Grisetti, NICP: Dense normal based point cloud registration, in: IEEE International Conference on Intelligent Robots and Systems, Vol. 2015-Decem, 2015, pp. 742–749.
[41] J. Yang, H. Li, Y. Jia, Go-icp: Solving 3d registration efficiently and globally optimally, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1457–1464.
[42] B. Yang, Z. Dong, F. Liang, Y. Liu, Automatic registration of large-scale urban scene point clouds based on semantic feature points, ISPRS Journal of Photogrammetry and Remote Sensing 113 (2016) 43–58.
[43] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J. Leonard, Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age, IEEE Transactions on Robotics 32 (6) (2016) 13091332.
[44] J. Nieto, T. Bailey, E. Nebot, Scan-SLAM: Combining EKF-SLAM and scan correlation, Springer Tracts in Advanced Robotics 25 (2006) 167–178.
[45] A. Das, M. Diu, N. Mathew, C. Scharfenberger, J. Servos, A. Wong, J. S. Zelek, D. A. Clausi, S. L. Waslander, Mapping, planning, and sample detection strategies for autonomous exploration, Journal of Field Robotics 31 (1) (2014) 75–106.
[46] M. Magnusson, A. Lilienthal, T. Duckett, Scan registration for autonomous mining vehicles using 3D-NDT, Journal of Field Robotics 24 (10) (2007) 803–827.
[47] A. Segal, D. Haehnel, S. Thrun, Generalized-ICP, in: Robotics: Science and Systems, Vol. 5, 2009, pp. 168–176.
[48] F. Pomerleau, F. Colas, R. Siegwart, S. Magnenat, Comparing icp variants on real-world data sets, Autonomous Robots 34 (3) (2013) 133–148.
[49] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, C. Cadena, Segmatch: Segment based loop-closure for 3d point clouds, arXiv preprint arXiv:1609.07720.
[50] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, A. J. Davison, Slam++: Simultaneous localisation and mapping

at the level of objects, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1352–1359.
[51] E. Fernández-Moral, P. Rives, V. Arévalo, J. González-Jiménez, Scene structure registration for localization and mapping, in: Robotics and Autonomous Systems, 2016, pp. 649–660.
[52] E. Fernández-Moral, W. Mayol-Cuevas, V. Arévalo, J. Gonzalez-Jimenez, Fast place recognition with plane-based maps, in: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, 2013, pp. 2719–2724.
[53] K. Georgiev, R. T. Creed, R. Lakaemper, Fast plane extraction in 3d range data based on line segments, in: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE, 2011, pp. 3808–3815.
[54] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM 24 (6) (1981) 381–395.
[55] J. Forsberg, U. Larsson, A. Wernersson, Mobile robot navigation using the range-weighted hough transform, IEEE Robotics & Automation Magazine 2 (1) (1995) 18–26.
[56] G. Pandey, S. Savarese, J. R. McBride, R. M. Eustice, Visually bootstrapped generalized icp, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 2660–2667.
[57] K. Lenac, A. Kitanov, R. Cupec, I. Petrović, Fast planar surface 3d slam using lidar, Robotics and Autonomous Systems 92 (2017) 197–220.
[58] N. Mellado, D. Aiger, N. J. Mitra, Super 4PCS fast global pointcloud registration via smart indexing, Computer Graphics Forum 33 (5) (2014) 205–215.
[59] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, T. Funkhouser, 3dmatch: Learning the matching of local 3d geometry in range scans, arXiv 1603.
[60] M. Li, X. Gao, L. Wang, G. Li, Automatic registration of laser-scanned point clouds based on planar features, in: 2nd ISPRS International Conference on Computer Vision in Remote Sensing (CVRS 2015), Vol. 9901, International Society for Optics and Photonics, 2016, p. 990103.
[61] Toward Object-based Place Recognition in Dense RGB-D Maps.
[62] A. Sehgal, D. Cernea, M. Makaveeva, Real-time scale invariant 3d range point cloud registration, in: International Conference Image Analysis and Recognition, Springer, 2010, pp. 220–229.
[63] S. Rusinkiewicz, M. Levoy, Efficient variants of the ICP algorithm, in: Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM, 2001, pp. 145–152.
[64] A. Gressin, C. Mallet, N. David, Improving 3D Lidar Point Cloud Registration Using Optimal Neighborhood Knowledge, ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences I-3 (September) (2012) 111–116.
[65] Z. Zhang, Iterative point matching for registration of free-form curves and surfaces, International journal of computer vision 13 (2) (1994) 119–152.
[66] J. S. Vitter, Faster Methods for Random Sampling, Commun. ACM 27 (7) (1984) 703–718.
[67] R. B. Rusu, Semantic 3D object maps for everyday manipulation in human living environments, Ph.D. thesis, Technische Universitt Mnchen, Diss., 2009 (2009).
[68] F. Donoso, K. Austin, P. McAree, How do ICP variants perform when used for scan matching terrain point clouds?, Robotics and Autonomous Systems 87 (2017) 147 – 161.
[69] J. Elseberg, S. Magnenat, R. Siegwart, N. Andreas, Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration, Journal of Software Engineering for Robotics (JOSER) 3 (1) (2012) 2–12.
[70] L. He, X. Wang, H. Zhang, M2DP: A novel 3D point cloud descriptor and its application in loop closure detection, in: Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, IEEE, 2016, pp. 231–237.
[71] K. Klasing, D. Althoff, D. Wollherr, M. Buss, Comparison of surface normal estimation methods for range sensing applications, in: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, IEEE, 2009, pp. 3206–3211.
[72] M. L. Tazir, P. Checchin, L. Trassoudaine, Color-based 3D Point Cloud Reduction, in: the 14th International Conference on Control, Automation, Robotics and Vision, ICARCV, 2016, pp. 1–7.
[73] T. Wiemann, M. Mrozinski, D. Feldschnieders, K. Lingemann, J. Hertzberg, Data Handling in Large-Scale Surface Reconstruction, in: 13th International Conference on Intelligent Autonomous Systems, 2014, pp. 1–12.
[74] D. Arthur, S. Vassilvitskii, K-means++: The Advantages of Careful Seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '07, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007, pp. 1027–1035.
[75] R. Tibshirani, G. Walther, T. Hastie, Estimating the number of clusters in a data set via the gap statistic, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63 (2) (2001) 411–423.
[76] P. J. Huber, E. M. Ronchetti, Robust statistics, John Wiley & Sons, 2009.
[77] P. J. Huber, Robust statistics, Wiley, New York, New York, 1981, Ch. The Basic Types of Estimates, pp. 45–67.
[78] A. Bjorck, Numerical methods for least squares problems, Siam, 1996.
[79] E. Puttonen, M. Lehtomäki, H. Kaartinen, L. Zhu, A. Kukko, A. Jaakkola, Improved sampling for terrestrial and mobile laser scanner point cloud data, Remote Sensing 5 (4) (2013) 1754–1773.