J Real-Time Image Proc manuscript No.
(will be inserted by the editor)

Rubén Usamentiaga · Daniel F. García · Julio Molleda

Efficient registration of 2D points to CAD models for real-time applications

Received: date / Revised: date

Abstract Efficient registration is a major challenge for real-time machine vision applications. Modern acquisition hardware can produce data at extremely high rates. Thus, efficient registration algorithms are required to align data to reference models in order to detect deviations and take correcting actions if needed. In this work, an efficient registration procedure of 2D points to CAD (Computer-aided design) models is proposed. Recent developments in the field are reviewed and evaluated in terms of accuracy, speed and robustness. Efficient algorithms are proposed for the most computationally expensive parts of the registration, including an estimation of the rigid transform, a calculation of the closest point to geometric primitives, and an estimation of the surface normal. Furthermore, a novel primitive caching procedure is proposed that, when combined with an R-tree, greatly improves the execution speed of the registration. The result is a very accurate registration procedure, since geometric primitives are treated analytically with no point sampling required. At the same time, the proposed procedure is robust, very fast, and can achieve the correct registration in less than one millisecond.

1 Introduction

Surface registration is a fundamental task in computer vision that is required as an intermediate, but crucial, step in many different applications. The goal is to optimally align one shape with another. The two shapes, each with its own coordinate system, are usually called the model and the data. The objective is to find the transformation that optimally aligns, or registers, the data with respect to the model [1]. Registration is used in a wide variety of applications, such as reverse engineering [2,3], cultural heritage [4], augmented reality [5], robotics [6,7], and object recognition [8].

Registration can be divided into several types, depending on the kind of data used for alignment [9]. For example, in model reconstruction, the goal is to create a complete object model from different views. In this case, both the model and the data are a set of points acquired from a surface reconstruction sensor, such as a structured light sensor [10]. Due to the limited field of view of the sensor and to occlusions, more than one view is required to represent the entire object. In this case, registration is used to align the partially overlapping views in order to build a complete object model. In multimodal registration, the goal is also to align several views of the same object. However, in this case, the information is acquired from different sources, such as MRI and CT scans [11]. In model fitting, registration is applied between a view of the object and a reference model. In this case, the data is a set of points representing a partial view of the object, and the object is a known CAD (Computer-aided design) model of the actual object.

Registration methods can also be classified according to different criteria. In [12], two main types are identified: coarse and fine registration methods. The difference is that fine registration methods require initial information and achieve much more accurate results. In general, coarse methods are used as the first step of the fine registration methods. These two types are further divided into different groups according to other aspects, such as the minimization distance, the method used to compute the transformation, and the estimation of the correspondences.

Most of the registration methods are based on the seminal Iterative Closest Point (ICP) algorithm [13,14]. The ICP algorithm estimates point correspondences between the data and the model. Originally, correspondences were estimated by searching for the closest point in the model to each point in the data. This is the point-to-point approach [13]. Another alternative is to calculate the projection of each point in the data onto the

Rubén Usamentiaga · Daniel F. García · Julio Molleda
Department of Computer Science and Engineering, University of Oviedo. Campus de Viesques, Gijón 33204, Asturias, Spain
Tel.: +34-985-182626
Fax: +34-985-181986
E-mail: rusamentiaga@uniovi.es

tangent plane at a model surface point located at the intersection of the normal vector of the data point. This is called the point-to-model approach [14]. Once point correspondences are estimated, ICP computes a geometric transformation between them. The standard algorithm computes a rigid transformation. However, recent developments in the field apply ICP to deformable registration [15]. Finally, the data is transformed by applying the computed transformation. The process is iterated until convergence. ICP iteratively improves the putative correspondences. It was proven that this algorithm is guaranteed to converge monotonically to a local solution of the problem. However, a global solution is only achieved when starting from a close solution. Otherwise, global convergence is not guaranteed. Therefore, an initial coarse registration needs to be applied to the data before ICP.

Since the introduction of the ICP algorithm many variants have been proposed. These variants improve the algorithm in terms of robustness [16–19], accuracy [20, 21], and speed [22–25], using different strategies.

Research on registration is mainly focused on 3D shapes. Most of these works can also be applied to 2D data, as it can be interpreted as a particular case by considering a zero z axis. However, these approaches create inefficient solutions, with inadequate algorithms for the computation of distances, correspondences and transformations.

Registration for 2D data is a required step in different applications. In the case of shape inspection, the observation by a camera of projected light onto the section of an object must be aligned with the model. The objective is to find deviations from the model, and detect defects. Complex inspection systems use several light projectors, generally laser stripes, and multiple cameras to avoid occlusions and to obtain a complete section of the object. Calibration is not a valid solution for registration in this type of applications. The movement of the inspected object is generally affected by vibrations, which change the position where the laser stripes are projected onto the object. Moreover, the inspected object generally moves very quickly, which requires very fast acquisition and processing modules. Modern cameras include a laser stripe extraction process [26]. This greatly reduces the bandwidth between the camera and the computer and the execution time required to extract laser stripes. This results in cameras that are able to provide hundreds of frames per second at megapixel resolution. When this resolution is slightly reduced, the obtained frame rate can be higher than 1000 fps. Therefore, the registration procedure must be extremely efficient to meet real-time constraints. In this work, the term "real-time" is interpreted in the signal processing sense [27], that is, based on the idea of completing the processing in the time available between successive input samples. Thus, the processing module should be fast enough to produce the results before the next frame is acquired.

In this work, an efficient registration for 2D is proposed. In particular, an efficient variant of the ICP algorithm is proposed for the registration of a set of points in two-dimensional Cartesian space to 2D CAD models defined using a set of geometric primitives. The main contributions of this paper are summarized next:

– An efficient, robust and accurate procedure is proposed that can align data obtained from standard structured light sensors in less than one millisecond, significantly faster than most commonly-used ICP variants. The goal of the proposed procedure is to produce very accurate alignment between the points and the model, while being fast enough to be used in a real-time application. Accuracy is an indispensable requirement for machine vision applications designed to take action based on the information inferred from the images. Execution speed is also of utmost importance because fast registration makes it possible to use the aligned points to measure the shapes of the objects during industrial manufacturing, and to take correcting actions if necessary.
– Recent developments in the field are reviewed and their applicability to this problem is evaluated in terms of accuracy, speed and robustness. Robustness variants in terms of outliers rejection are analyzed, and their impact on performance is evaluated.
– Efficient algorithms are proposed for the most computationally demanding tasks, including the estimation of the surface normal, the estimation of the rigid transform, and the computation of the closest point for the most common geometric primitives. These efficient algorithms are specially designed for real-time applications, and have a huge impact on the execution speed of the registration algorithm. These algorithms are used to compute the correspondences and during the proposed initial coarse registration.
– A comparison is carried out between a numerical approach using a discrete version of the models and fast kd-trees, and an analytical approach considering the geometric primitives of the CAD model and R-trees. Standard variants of the ICP algorithm are based on points, but this work proposes a novel procedure based on the geometric primitives of the model that can achieve much better accuracy with minimum execution times.
– A novel primitive caching method is proposed to further reduce the computational demands of the algorithm, which is also applicable to 3D. This process is accelerated using an R-tree. This spatial structure is designed for points, but in this case it is successfully adapted to be used with geometrical primitives.

The effects of the proposed optimization techniques are discussed, and compared with a numerical registration method using kd-trees. Extensive tests are carried out to evaluate the performance of the proposed procedure and the alternatives analyzed. Synthetic data is used in the tests in order to evaluate the performance of different strategies accurately, in terms of accuracy, speed and robustness. Moreover, the registration procedure is applied to an industrial application where efficient and real-time registration is required: a rail inspection system.

The remainder of this paper is organized as follows. Section 2 compares the numerical and the analytical approach; Section 3 presents the optimization of the analytical approach to reduce the execution speed and improve the robustness while maintaining the accuracy; Section 4 discusses the results obtained with synthetic and real data; and finally, Section 5 reports conclusions.

2 Numerical and analytical approach

The ICP algorithm is designed to deal with two sets of points or point clouds. However, a CAD model is defined in terms of basic geometric primitives, mainly line segments and circular arcs. Two different approaches can be applied for registration: a numerical approach where a discrete version of the model is used, or an analytical approach where the geometric primitives that define the model are used for registration.

2.1 Numerical approach

In order to apply the numerical approach, a set of points describing the surface of the model must be calculated. This process can be applied offline, thus it does not intrude upon the registration process. A CAD model is described in terms of geometric primitives. Therefore, a discrete version of the model can be obtained by sampling the primitives at a specific resolution. The geometric primitives form an ordered closed shape, that is, the final point of a primitive is the initial point of the next primitive. Therefore, the distance remaining while sampling one primitive is added to the next. In this way, it is possible to obtain a discrete version where all the points are at the same arc length distance from each other, producing an accurate description of the model.

In the numerical approach, a naive approach to compute the closest points from the data to the model is to compare the squared distances from all the points in the discrete version of the model in a loop.

2.2 Analytical approach

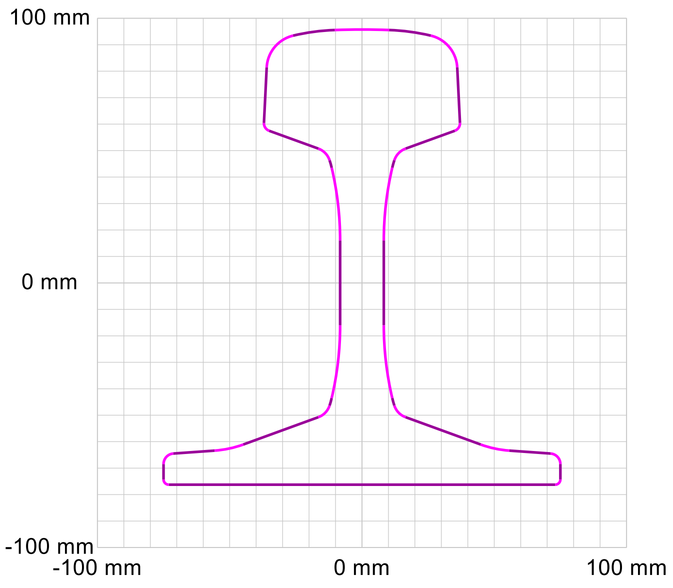
The analytical approach requires the definition of appropriate operators to compute the closest point. The estimation of correspondences in the registration process requires the computation of the closest point in the model from each point in the data. The closest point in the model is the closest point in any geometric primitive. Thus, a possible solution is the computation of the closest point in all geometric primitives. Then, the final closest point is selected from this set of points. Considering the model composed of line segments and circular arcs, a naive approach can be applied to the calculation of the closest point. In the case of a line segment, the closest point in the segment to a specific point is the intersection of the line segment with the line perpendicular to this segment that passes throughout the point. When the intersection does not lie onto the segment, the closest point is either the initial or the final point, the closest one. In the case of a circular arc, the process is similar. The closest point in the circular arc to a specific point is the intersection of the circle with the line that passes throughout the point and the center of the circular arc, but only when this point is inside the circular angle of the arc. Otherwise, it is either the initial or the final point of the arc, the one closest to it.

2.3 Registration

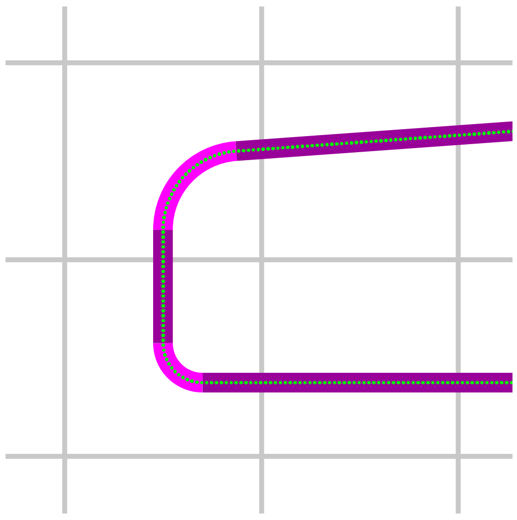
In order to evaluate the performance of the registration, a standard 2D model and a synthetically-generated point cloud based on the model are used. The model can be seen in Fig. 1a. It is a model of a rail defined in the UNE EN13674-1 standard, called 60E1. The model is composed of line segments and circular arcs that can be distinguished with different colors in the figure. Fig. 1a shows a detail of the point cloud used for tests. The point cloud contains 2708 points with an arc length distance of 0.25 mm between them.

The reason for using synthetic data for the initial performance evaluation is that the correct transform is known exactly, and the performance of different strategies can be assessed accurately. Two different metrics are used to assess the registration: average distance and average execution time. The average distance is calculated from the points after registration, to the model (calculated analytically). This metric allows for objective comparisons of the registration strategies in terms of accuracy. The second metric is the average execution time required by the registration. This value is calculated as an average of 100 experiments. All reported running times are for a C++ implementation running on an Intel Core i7 4770 running at 3.4 GHz with 16 GB of RAM. The implementation is compiled for x64 and the data type used to store the coordinates of the points is double-precision floating-point. The average execution time allows for objective comparisons of the registration strategies in terms of speed.

The data used for tests is randomly transformed before the registration, applying the same translation and a rotation. This transformation changes the position of the points. The goal of the registration is to align the data back to the model. These initial tests do not apply an initial coarse registration before ICP, because the objective is to compare the raw performance of the strategies.

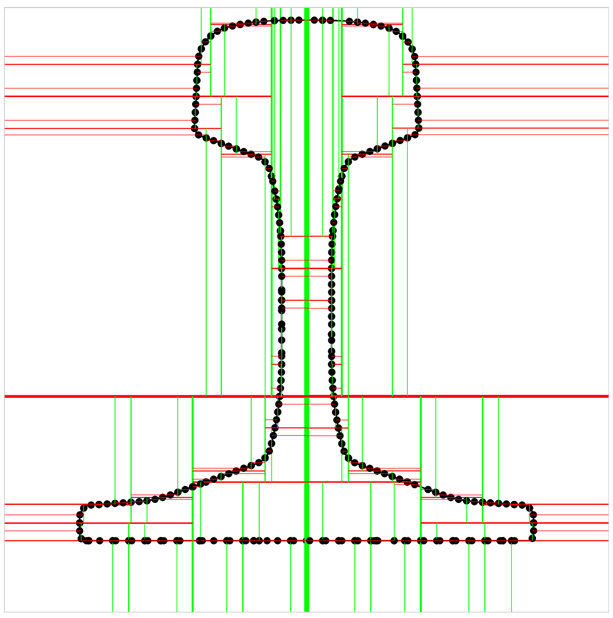


(a)

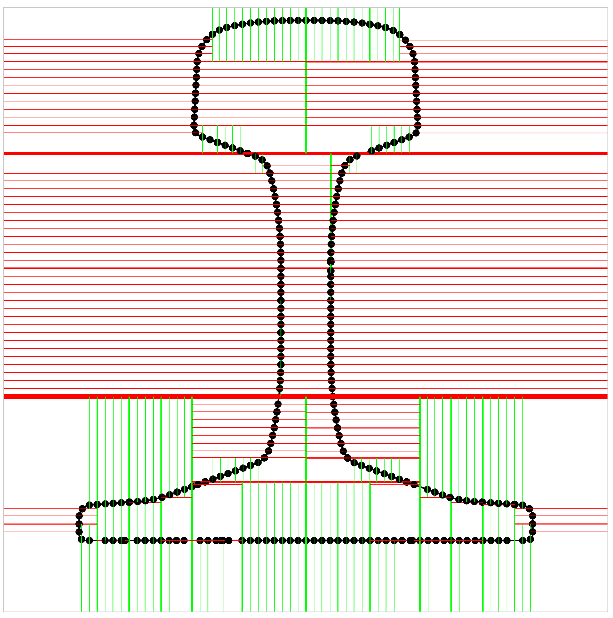


(b)

Fig. 1 Model and point cloud used for tests. (a) Model of rail 60E1. (b) Detail of point cloud in the left foot of the rail



(a)



(b)

Fig. 2 Spatial partitioning using kd-tree. (a) Conventional partitioning. (b) Partitioning based on the variance. Green lines indicate the limits of vertical partitions and red lines the limits of horizontal partitions.

Thus, the transformation applied to the data is small in terms of distance (only a translation of 5 mm), so that the registration does not fall into a local solution and is able to reach the global optimum.

Two convergence criteria are used for the registration tests. Firstly, that the distance between the data and the model is below 1e-6 mm. Secondly, that the distance from current correspondences to the model is worse than in the previous iteration, that is, that the registration is not improving. When any of these two conditions are met, the registration finishes. The rigid transform between the points and their correspondences in the model is estimated using a standard approach based on the cross-dispersion matrix and SVD [28].

The first two experiments compare the performance of the numerical and the analytical approach. For the numerical approach the correspondences are estimated using the naive method described in Section 2.1. In this first test, the discrete version of the model contains 6769 points with an arc length distance of 0.1 mm between them. For the analytical approach, the correspondences are estimated using naive method described in Section 2.2. The results of the numerical approach were an average distance of 0.0078 mm with an average execution time of 222.71 ms in 13 iterations. The results of the analytical approach were an average distance of 8.53E-07 mm with an average execution time of 323.41 ms in 35 iterations. Both the numerical and the analytical approaches are very slow, as they are implemented inefficiently. However, the relevant result is that the numerical approach provides much lower accuracy than the analytical approach. The numerical approach is only able to improve the registration for 13 iterations, and the final average distance is far from the convergence objective of 1e-6 mm. The numerical approach converges in fewer iterations than the analytical approach but with worse alignment. This result is not unexpected since sampling a continuous function leads to errors.

In order to improve the registration based on the numerical approach, a straightforward solution is to increase the resolution of the discrete version of the model. Improving the resolution of the discrete version of the model to 67689 points with an arc length distance of 0.01 mm does indeed reduce the average distance to 0.0023 mm in 19 iterations. However, the execution time increase dramatically to 3299.71 ms. Therefore, a small improvement in accuracy is obtained with a large increase of execution time.

The problem with the numerical approach is the naive implementation of the nearest-neighbor search. Exploring every possible pairing of points is $O(nm)$ where n and m are the number of points in the respective data and model. A much more efficient approach is to employ spatial data structures, which reduce the complexity in the order of $O(n \log m)$. Spatial data structures that partition space allow efficient access to the stored elements via positional queries [29]. Therefore, in order to improve the execution speed of the numerical approach, a kd-tree, which is a data structure for organizing points in a space [30], can be used to store the discrete version of the model. It is important to use a space partitioning strategy suitable to the data stored, as it has a strong influence on the performance [31]. Conventional kd-trees apply a partition axis selection where each axis is selected alternatively. This creates an inefficient binary space partitioning, as can be seen in Fig. 2a. Much more efficient in this case is a space partitioning based on the variance. The result is shown in Fig. 2b. As can be seen, by using this strategy the points forming a horizontal line are only divided with vertical partitions, whereas the points forming a vertical line are only divided with horizontal partitions. This gives much more efficient partitioning which in turn greatly increases the performance of the nearest-neighbor search. This type of efficient partitioning increases the most computationally expensive part using a kd-tree: the initialization or insertion of the points in the hierarchical data structure. However, in this case this process is carried out offline and does not affect the execution speed. In this work, the nanoflann library is used for kd-tree, which is an optimized version for 2D or 3D point clouds of the flann library [32].

The results with the numerical approach implemented using kd-trees are no more accurate than those using the naive implementation for the same resolution of the discrete model. However, the execution speed is reduced drastically. When using the discrete version of the model that contains 6769 points (resolution 0.1 mm), the execution speed is reduced to 20.22 ms. The registration using the higher resolution model with 67689 points (resolution 0.01 mm) requires 38.32 ms, compared with the 3299.71 of the naive implementation. This low execution speed makes it possible to use discrete models with even better resolution, as further increasing the resolution of the model produces better accuracy with low overhead. Improving the resolution of the discrete version of the model to 135378 points with an arc length distance of 0.005 mm reduces the average distance to 0.0012 mm in 21 iterations. The execution time in this case increases to 46.39 ms. The results of these experiments indicate that using kd-trees greatly improves the performance of the numerical approach in terms of execution speed. However, the improvement in terms of accuracy is low, even when the discrete model is sampled with very high resolution. The obtained accuracy is still much lower than the accuracy obtained using the analytical approach.

The analytical approach provides much more accurate results than the numerical approach, but is also much more computationally expensive. A tradeoff appears between accuracy and execution speed. The accuracy with the numerical approach cannot really be improved. Further increasing the resolution of the discrete model greatly increases the required memory for very low improvement in accuracy. On the other hand, the analytical approach, although slow, still has room for improvement by optimizing different aspects. Moreover, it is the only approach that can achieve the accuracy indicated in the first convergence condition. Therefore, the rest of this paper will develop the analytical approach by proposing different strategies that improve the performance of this approach in terms of execution speed and robustness.

3 Optimization of the analytical approach

Computing the correspondences is the most computationally expensive part of the registration process. Therefore, improving the computation of the closest point to line segments and circular arcs is essential for improving the execution speed of the algorithm using the analytical approach. Other alternatives for improving the registration speed are explored in this section, such as the estimation of the rigid transform. The resulting execution speed is evaluated with the test data.

3.1 Efficient estimation of the rigid transformation in 2D

During the registration process, a rigid transformation between the data and the model must be estimated for each iteration. This is a computationally demanding mathematical procedure that is applied between thousands of points. Thus, it is important to make this step more efficient in order to improve the registration performance.

Rigid transformation is one of the classes of geometric transformations. These classes can be described in terms of those elements or quantities that are preserved or invariant, such as the distance or the angle [33]. A rigid transformation preserves the Euclidean distance; this is why it is also called Euclidean transformation. This type of transformation is a composition of translations, rotations, reflections and compositions of these. However, reflections are commonly excluded from the definition by imposing that the transformation also preserve the handedness.

Different methods can be used to estimate a rigid transformation between two point clouds. The most common are based on computing the singular value decomposition (SVD) of the cross-dispersion matrix or quaternions [34]. These methods are generally applied to 3D data, but can be adapted to 2D.

Considering a set of n points $\mathcal{P} = \{p_1, p_2, \dots, p_n\}$, and $\mathcal{Q} = \{q_1, q_2, \dots, q_n\}$ in $\mathbb{R}^2$, where $p_i = (p_{ix}, p_{iy})^T$ represents the 2D coordinates of the i-th point in $\mathcal{P}$, a rigid transformation composed of a rotation R and a translation t provides a mapping between these two datasets as (1).

$$\mathcal{Q} = \mathcal{P}R + t \tag{1}$$

Solving for the optimal rigid transformation (R and t) that maps $\mathcal{P}$ onto $\mathcal{Q}$ requires minimizing E, the least squares error criterion (2).

$$E = \sum_{i=1}^{n} |\mathcal{Q} - \mathcal{P}R - t|^2 \tag{2}$$

The value of t minimizing E must satisfy (3).

$$0 = \frac{\partial E}{\partial t} = -2\sum_{i=1}^{n} |\mathcal{Q} - \mathcal{P}R - t| \tag{3}$$

Therefore, t can be obtained from (4), where $\bar{p}$ and $\bar{q}$ are the centroids of $\mathcal{P}$ and $\mathcal{Q}$, calculated using (5).

$$t = \bar{q} - R\bar{p} \tag{4}$$

$$\bar{p} = \frac{1}{n}\sum_{i=1}^{k} p_i,\ \bar{q} = \frac{1}{n}\sum_{i=1}^{k} q_i \tag{5}$$

Introducing the centered points $\mathcal{P}^z = \{p_1^z = p_1 - \bar{p}, p_2^z = p_2 - \bar{p}, \dots, p_n^z = p_n - \bar{p}\}$, and $\mathcal{Q}^z = \{q_1^z = q_1 - \bar{q}, q_2^z = q_2 - \bar{q}, \dots, q_n^z = q_n - \bar{q}\}$ in (2) yields (6).

$$E = \sum_{i=1}^{n} |\mathcal{Q}^z - \mathcal{P}^z R|^2 \tag{6}$$

The estimation of R that minimizes E in (6) is known as the orthogonal Procrustes problem [35], because $R^T R = I$, i.e., the rotation matrix is orthonormal.

The standard method for estimating R for 3D points is to calculate the cross-dispersion matrix [28], which is defined as (7).

$$C = \frac{1}{n}\sum_{i=1}^{n} \mathcal{P}^z \mathcal{Q}^{zT} \tag{7}$$

The matrix C is decomposed using SVD as (8).

$$C = USV^T \tag{8}$$

The rotation matrix R is finally calculated using (9). Extended details about the mathematical procedure that leads to this equation are given in [36].

$$R = UV^T \tag{9}$$

The calculation of the translation t is obtained by substituting the value of R in (4).

In 2D, the estimation of R can be greatly simplified. The rotation matrix is defined by θ, the angle of rotation. The rotation of point p_i^z by the angle θ gives (10).

$$Rp_i^z = \begin{pmatrix} cos(\theta)p_{ix}^z - sin(\theta)p_{iy}^z \\ sin(\theta)p_{ix}^z + cos(\theta)p_{iy}^z \end{pmatrix} \tag{10}$$

Substituting (10) in (6) gives a function where E only depends on θ. Minimizing E involves taking the derivative with respect to θ and solving for θ when the derivative is zero. The result is (11).

$$\theta = \tan^{-1}\left(\frac{\sum_{i=1}^{n}(p_{ix}^z q_{iy}^z - p_{iy}^z q_{ix}^z)}{\sum_{i=1}^{n}(p_{ix}^z q_{ix}^z + p_{iy}^z q_{iy}^z)}\right) \tag{11}$$

The angle θ can be also calculated from the coefficients of the cross-dispersion matrix, as (12) and (11) are equivalent.

$$\theta = \tan^{-1}\left(\frac{C_{12} - C_{21}}{C_{11} + C_{22}}\right) \tag{12}$$

The calculation of θ has a geometric interpretation by considering p_i^z and q_i^z as the coordinates of two vectors in $\mathbb{R}^2$. The dot product between these two vectors can be calculated using (13). The perp dot product between these two vectors, where one vector is replaced by its perpendicular vector, can be calculated using (14). These two vector products have equivalent algebraic and geometric definitions.

$$p_i^z \cdot q_i^z = |p_i^z||q_i^z|cos(\theta) = p_{ix}^z q_{ix}^z + p_{iy}^z q_{iy}^z \tag{13}$$

$$p_i^{z\perp} \cdot q_i^z = |p_i^z||q_i^z|sin(\theta) = p_{ix}^z q_{iy}^z - p_{iy}^z q_{ix}^z \tag{14}$$

Dividing (14) by (13) results in (15). Therefore, θ can be calculated using (16).

$$\frac{p_i^{z\perp} \cdot q_i^z}{p_i^z \cdot q_i^z} = \frac{|p_i^z||q_i^z|sin(\theta)}{|p_i^z||q_i^z|cos(\theta)} = \frac{sin(\theta)}{cos(\theta)} \tag{15}$$

$$\theta = \tan^{-1}\left(\frac{p_i^z \perp q_i^z}{p_i^z \cdot q_i^z}\right) = \tan^{-1}\left(\frac{p_{ix}^z q_{iy}^z - p_{iy}^z q_{ix}^z}{p_{ix}^z q_{ix}^z + p_{iy}^z q_{iy}^z}\right) \tag{16}$$

The rotation matrix R for θ is finally obtained from (17).

$$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{17}$$

The rotation matrix obtained in (17) is the same as the rotation matrix obtained in (9). However, the calculations required in (11) to obtain the rotation angle are more efficient than applying SVD to the cross-dispersion matrix. Therefore, the required execution time is greatly reduced, which is very important for time-constrained applications.

The registration of the test data with the proposed estimation of rigid transform reduces the execution time from 323.41 ms to 281.86 ms. This not a great reduction because the estimation of the rigid transform is not the most computationally demanding part of the registration. However, it is a step toward improving the global execution time of the algorithm.

3.2 Efficient closest point

3.2.1 Closest point to a line segment

The parametric form of a line defined by two points P_0 and P_1 is $P(t) = P_0 + td$ for $t \in \mathbb{R}$, where $d = (P_1 - P_0)$ is the direction vector. A line segment, or simply a segment, is a particular case of a line with $t \in [0, 1]$. In this case, t represents the fraction of distance along the whole segment. If $t < 0$, then $P(t)$ is outside the segment on the P_0 side, and if $t > 1$ then $P(t)$ is outside on the P_1 side [37].

Given a point P and a line $\mathcal{L}$, the closest point Q on the line $\mathcal{L}$ to P is the projection of P onto $\mathcal{L}$, as can be seen in Fig. 3. The vector $(P-Q)$ must be perpendicular to the direction vector d. Thus, the dot product of these two vectors must be zero, satisfying (18). Solving this equation for t gives (19).

$$0 = d \cdot (P - Q) = d \cdot (P - P_0 - td) \tag{18}$$

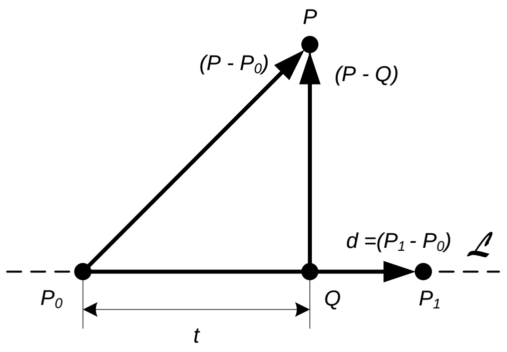


Fig. 3 Closest point to a line segment.

$$t = \frac{d \cdot (P - P_0)}{|d|^2} \tag{19}$$

In the case of the line segment $\mathcal{S}$, the projection of P onto $\mathcal{S}$ may lie onto the segment, behind the initial point (P_0), or ahead of the final point of the segment (P_1). These three possibilities depend on t. Therefore the closest point Q on a segment to a point P is obtained from (20).

$$Q = \begin{cases} P_0 & t \leq 0 \\ P_0\text{+td} & t \in (0, 1) \\ P_1 & t \geq 1 \end{cases} \tag{20}$$

The calculation of the closest point to a segment can be further optimized. The quantities $a = |d|^2$ (the denominator in (19)) and $b = d/|d|^2$ can be precomputed and stored with the information about the line segment. Then, in order to obtain the closest point, the numerator (n) in (19) is calculated first using (21). This is the dot product between the direction vector and the vector from the initial point of the segment to P. The resulting value of this product indicates an obtuse angle between the vectors when it is negative. In this case, the closest point is P_0. When the value of this dot product is greater than or equal to a (the denominator in (19)), the closest point is P_1, because the result of the division would be greater than or equal to one. Only when neither of these two conditions are satisfied, are further calculations required. Finally, the closest point is calculated using (22). This expression further improves the efficiency of the algorithm.

$$n = d \cdot (P - P_0) \tag{21}$$

$$Q = \begin{cases} P_0 & n \leq 0 \\ P_1 & n \geq a \\ P_0\text{+bn} & otherwise \end{cases} \tag{22}$$

3.2.2 Closest point to a circular arc

A circular arc, or simply an arc, is a portion of the circumference of a circle. It is defined by a central point, C, the radius, r, and the initial and final angles, α_0, and

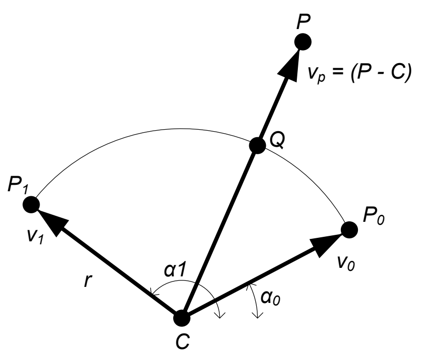


Fig. 4 Closest point to a circular arc.

α_1. The central point and the radius define the circle of which the arc is a part. Arcs can be clockwise or counterclockwise, but they can be mathematically treated the same by swapping the initial and final angles.

Angle normalization makes further calculations more simple. Any angle α can be normalized as $((\alpha \bmod 2\pi)+2\pi) \bmod 2\pi$. The resulting angle $\in [0, 2\pi]$. Then, if α_1 is less than α_0, 2π is added to α_1. This way the circular angle of the arc is $\alpha_1 - \alpha_0$.

Given a point P and a circle $\mathcal{C}$, the closest point Q on the circle $\mathcal{C}$ to P is calculated using the vector $v_p = (P-C)$, as can be seen in Fig. 4. This is the vector from C (the center of the circle) to P. The closest point is the unit vector of v_p multiplied by r (the radius) plus the center point, as shown in (23).

$$Q = \frac{v_P}{|v_P|} r + C \tag{23}$$

The point Q is also the closest point to an arc $\mathcal{A}$, but only when P is inside the circular angle of $\mathcal{A}$. This can be tested by calculating the angle of v_p using the expression $\alpha_P = tan^{-1}(v_{Py}/v_{Px})$ (calculated using `atan2` function). The point P is inside the circular angle of $\mathcal{A}$ when $\alpha_0 \leq \alpha_P \leq \alpha_1$. If α_P is normalized the same as α_1, the test expression can be simplified as $\alpha_P \leq \alpha_1$. The problem with this approach is that the computation of tan^{-1} is complex and heavy. A more efficient approach can be used.

An efficient solution to test if a P is inside the circular angle of $\mathcal{A}$ can be obtained using the perp dot product. The vectors from the center of the arc to the initial and final points of the arc can be precomputed as $v_0 = (\cos(\alpha_0), \sin(\alpha_0))$ and $v_1 = (\cos(\alpha_1), \sin(\alpha_1))$. The sign of the perp dot product between two vectors indicates, when positive, that the second vector is counterclockwise from the first, or clockwise when negative [38]. Therefore, P is inside the circular angle of $\mathcal{A}$ when $c_0 = v_0^{\perp} \cdot v_P > 0$ and $c_1 = v_1^{\perp} \cdot v_P < 0$. When the circular angle of $\mathcal{A}$ is reflex ($> \pi$), then P is inside when $c_0 > 0$ or $c_1 < 0$.

When the point is outside the circular angle, the closest point is either the initial or the final point of the arc, $P_0 = C + r(\cos(\alpha_0), \sin(\alpha_0))$, or $P_1 = C + r(\cos(\alpha_1), \sin(\alpha_1))$, which can be precomputed. Whether it is P_0 or P_1 can be determined by comparing the squared distances from P. A more efficient alternative is the computation of the dot product between the direction vector $b = (P1-P0)/|P1-P0|^2$ and $(P-P_0)$. When the result of the dot product is less or equal than 0.5, the closest point is P_0. Otherwise, it is P_1. The mathematical reasoning can be found in Section 3.2.1. This approach is slightly faster than the computation of the squared distances because b can be precomputed.

Finally, the closest point is calculated using (24), where $R(\mathcal{A})$ is a precomputed boolean that indicates if the circular angle of $\mathcal{A}$ is reflex. This expression to calculate the closest point to an arc is very efficient and does not require the computation of any computationally expensive trigonometric function.

$$Q = \begin{cases} \frac{v_P}{|v_P|} r + C & (c_0 > 0 \wedge c_1 < 0) \vee \\ & (R(\mathcal{A}) \wedge (c1 > 0 \vee c1 < 0)) \\ P_0 & (P - P_0) \cdot b \leq 0.5 \\ P_1 & otherwise \end{cases} \tag{24}$$

Applying the proposed procedures to calculate the closest point to the line segment and circular arcs of the model has a huge impact on the execution speed of the registration algorithm. The average execution speed is reduced to 22.58 ms. This is a speedup of ×14. Using the proposed efficient procedures, the execution speed of the analytical approach is similar to the one obtained with the numerical approach and kd-trees. However, the analytical approach provides much better accuracy.

3.3 R-trees

The computation of the closest point to the model requires the calculation of the closest point to each geometric primitive of the model. The final closest point is then obtained as the closest in terms of squared distance from this set of points. However, the spatial position of the point can be used to reduce the number of required tests. Only the close primitives of the points really need to be used for the closest point. The R-tree is an efficient spatial data structure that can be used for this purpose [39].

An R-tree is a hierarchical data structure that stores nodes representing an axis-aligned bounding box of arbitrary dimensions that can overlap. The primary use of this data structure is in geographic information systems. The search in an R-tree provides the nodes that contain a point in space. Therefore, in the case of the registration process it can be used to determine efficiently the closest geometric primitives of a point. Then, only these primitives need to be tested to calculate the closest point in the model.

In order to create the R-tree for the model, the following approach is proposed. First, a closed envelope region

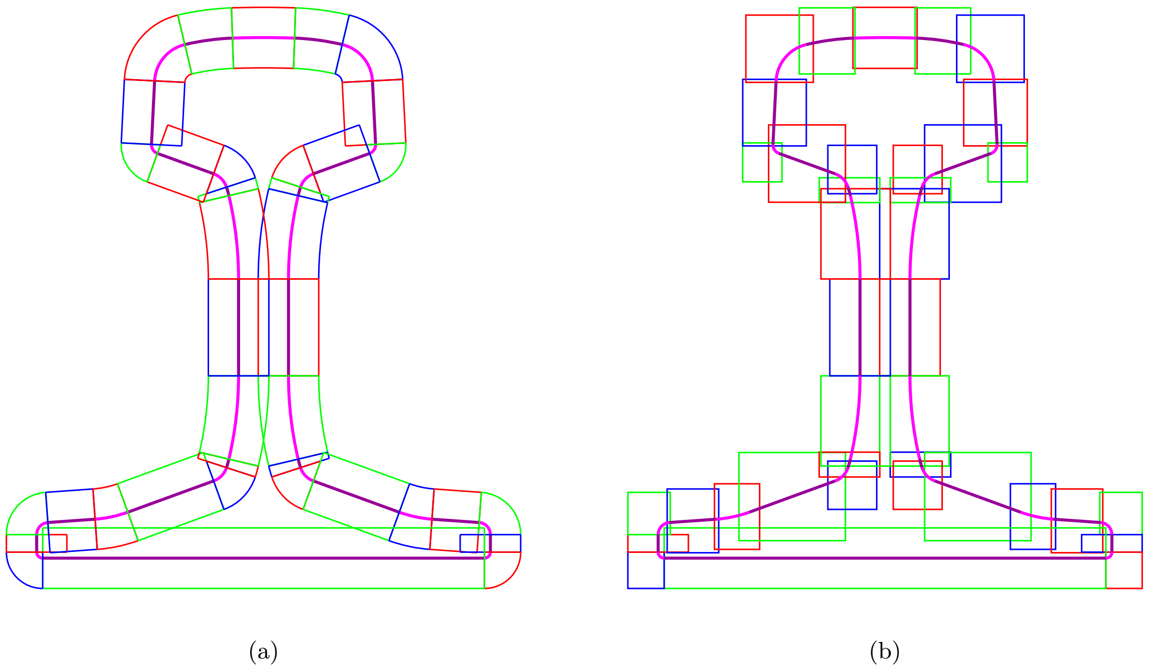


Fig. 5 Spatial partitioning using R-tree. (a) Envelope of geometric primitives using a distance threshold of 10 mm. (b) Bounding box of geometric primitives based on the envelopes regions.

around every geometric primitive is created. These envelope regions are created so that only relevant points closer than a distance threshold to the primitive are included. The envelope is restricted by the perpendiculars of the initial and final points of each primitive. The points before or after these points may be close to the primitive, but will be closer to the next or previous primitive of the model. In this way, the envelope region represents an area that can be used to identify the corresponding primitive to a point. The result can be seen in Fig. 5a. It is important to take into account that ICP is a fine registration method where data points are close to the model. Therefore, a low distance threshold can be used in most cases to create the envelope regions.

The envelope regions are good indications of which primitive is closer to a point. However, determining if a point is inside these regions is not computationally efficient. Therefore, the envelopes are transformed into rectangles. A rectangle is created for each envelope by calculating the bounding box of the envelope. The result can be seen in Fig. 5b. These rectangular regions are not as precise as the envelopes to determine which geometric primitive is closer, but they are much more efficient to work with. The coordinates of these rectangular regions are the coordinates with which the R-Tree is created.

In the worst-case scenario, a point can be inside multiple regions. This is the case shown in Fig. 6a using a distance threshold of 10 mm. In this example, the point is inside four regions. Therefore, in order to calculate the closest point of the model tests are required with the corresponding four geometric primitives. Fortunately, the most common case is when the point is contained inside only one region, as can be seen in Fig. 6b. In this case, the computation of the closest point is the computations of the closest point with a single geometric primitive.

Using the proposed R-tree creates a singular problem when there are outliers in data points. Some of these points could be located outside all the regions. In this case, the closest point can be calculated using the previous approach: iteration between all the geometric primitives. A tradeoff appears when selecting the most appropriate value for the distance threshold used to create the regions of the R-tree. A low value reduces the overlapping regions. Therefore, searching the closest geometric primitive to a point is more efficient. However, when using very small envelope regions, some points could be outside all the envelope regions, provoking a brute-force search.

Using an R-tree makes the computation of the correspondences much more efficient. The average execution speed is reduced to 6.76 ms for the registration of the test data.

3.4 Primitive caching

Caching accelerates the speed of conventional registration using ICP by storing a subset of model points that

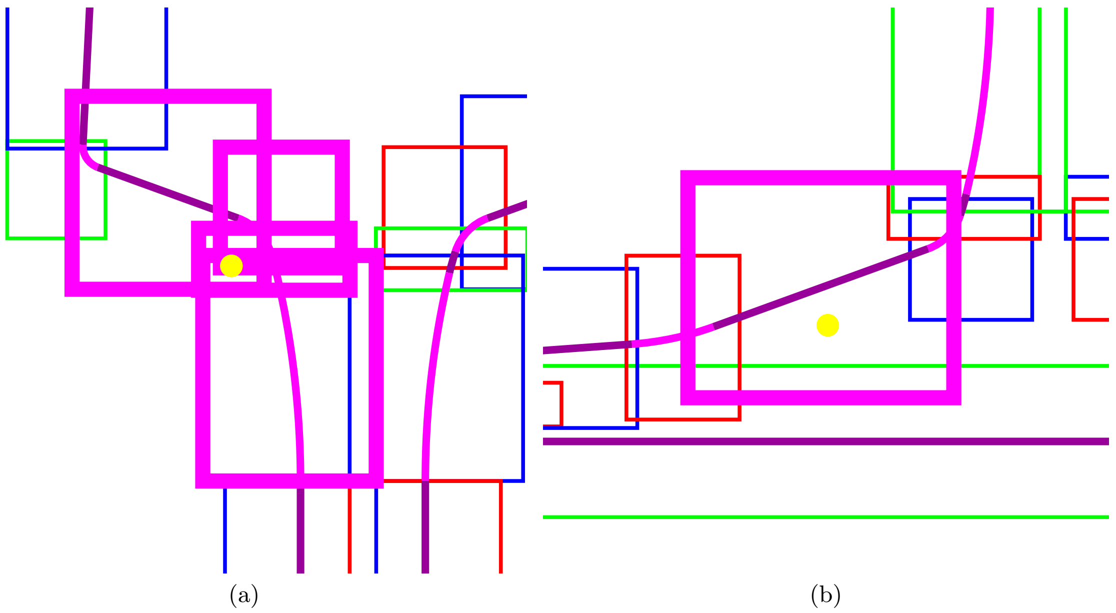


Fig. 6 Worst and best-case scenario when using R-tree to search for the closest geometric primitives. (a) Worst-case: the point is inside four regions. (b) Best and most common case: the point belongs to only one region so the closest point can be directly calculated from the corresponding geometric primitive.

are close to a data point at a previous iteration [40]. This work proposes a similar approach to further accelerate the proposed registration procedure. However, rather than point caching, in this case a primitive geometric caching procedure is proposed.

ICP is a fine registration method where the position of data points does not change significantly between iterations. Therefore, when a point is close to a geometric primitive at one iteration it is highly probable that the corresponding geometric primitive in the next iteration will be the same. This could be interpreted as temporal locality: if at one iteration a point has a particular corresponding geometric primitive, then it is likely that the same geometric primitive will be the corresponding primitive for the same point at the next iteration.

The proposed primitive caching procedure is as follows. Initially, a corresponding primitive is assigned to each data point: the geometric primitive that contains the closest point in the model to the data point. This initial process is accelerated using an R-tree. When the estimation of correspondences is required, only the closest point in the cached corresponding primitive is calculated. This is how the initial corresponding points are calculated. The change in the positions of the points due to the alignment with respect to the model can make a cached corresponding primitive become invalid. This can be detected by comparing the obtained corresponding point with the initial and final point of the cached primitive. If this is not the case, the initial corresponding point is correct, but when this case occurs, a new primitive must be assigned as the corresponding primitive for that point. If the corresponding point is the initial

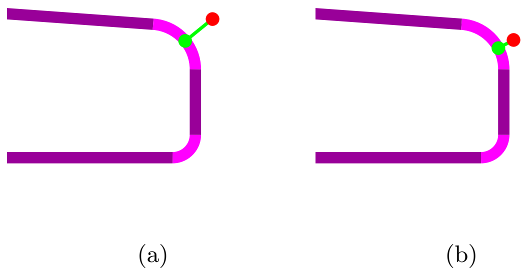


Fig. 7 Best and most frequent scenario for primitive caching. (a) Initial correspondences. (b) New iteration: the closest point is calculated directly from the cached corresponding primitive.

point of the current cached primitive, a new corresponding point is calculated using the previous primitive. The resulting point could be the same, which indicates that the closest point is the intersection of the primitives (initial point of the current primitive and final point of the previous primitive). If the corresponding point is the final point, the same procedure is applied using the next primitive. This process is repeated while the corresponding point is either the initial or the final point of the current primitive and that point has not been obtained before. This iterative process ends with one valid corresponding point and updated corresponding primitive that will be cached for the next iteration of the registration.

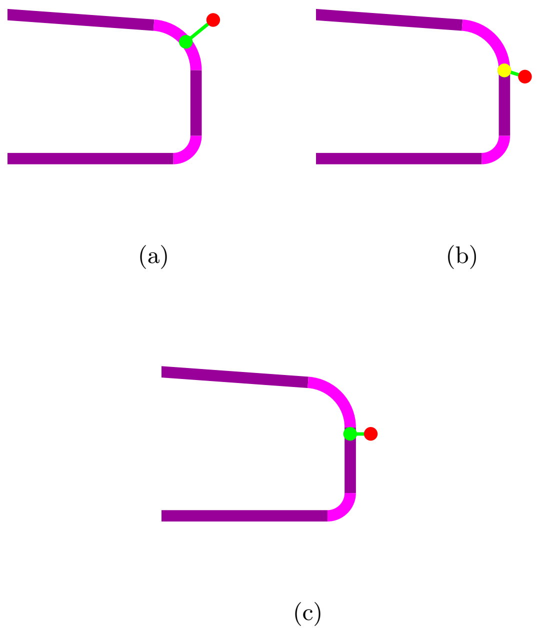


Fig. 8 Primitive caching when updating is required. (a) Initial correspondences. (b) New iteration: the corresponding point is the final point of the cached corresponding primitive. (c) The cached corresponding primitive is updated and the correct closest point is calculated.

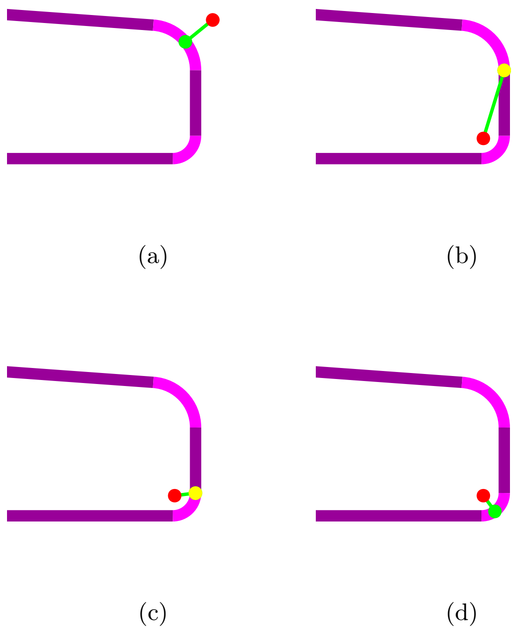


Fig. 9 Primitive caching when more than one update is required. (a) Initial correspondences. (b) New iteration: the corresponding point is the final point of the cached corresponding primitive. (c) The new corresponding point is the final point of updated primitive. (d) The cached corresponding primitive is updated and the correct closest point is calculated.

The most frequent case is when the obtained initial corresponding point is neither the initial nor the final point of the corresponding primitive. This is the case shown in Fig. 7. In the new iteration, the closest point is calculated directly from the cached corresponding primitive.

Much less frequently, the corresponding point is the initial or final point of the cached corresponding primitive. When this occurs, the corresponding primitive must be updated. This is the case shown in Fig. 8. In the new iteration, it is detected that the cached corresponding primitive is no longer valid. Thus, it needs to be updated to obtain the correct corresponding point.

Under very specific conditions, the correct corresponding primitive is before or after two or more primitives than the cached primitive. This is the case shown in Fig. 9. In this case, the cached corresponding primitive needs to be updated twice to obtain the correct corresponding point. This case only occurs when a large displacement is applied to a point or when geometric primitives are very short.

Primitive caching further reduces the execution speed of the registration. Using the proposed caching procedure, the average execution speed is reduced to 1.57 ms. Therefore, from the original naive analytical approach with an execution time of 323.41 ms, the proposed efficient registration has obtained a speedup of $\times 304$. The improvement in execution speed has been carried out with no sacrifices in accuracy. The resulting procedure reaches the same result, but in only a tiny portion of time. Moreover, the result obtained is not only much more accurate than the numerical approach, but also much faster than the optimized version using kd-trees.

3.5 Improving robustness

The efficient registration procedure proposed is not yet complete. It is a valid test to compare different strategies for the computation of the fine registration. However, robust registration requires an appropriate initial coarse registration that approximates the data to the model so that the ICP does not get stuck in a local minimum. Moreover, considering all the correspondences between the data and the model as valid does not produce a good registration in the presence of noise or outliers in the data. Incorrect correspondences bias the obtained result, preventing the registration from finding the correct alignment.

3.5.1 Initial coarse registration

For the initial coarse registration, a simple yet effective strategy is proposed: alignment of centroids and orientation. Centroid and orientation for the model are calculated offline. However, the equivalent features for the data must be calculated during the initial coarse registration. The centroid calculation is very simple, but the orientation requires a procedure based on principal component analysis to estimate the surface normal. Next, an efficient procedure to estimate surface normals in 2D is described. The procedure is described for a generic case where the surface normal is calculated for a point with a local neighborhood. The orientation of the data is calculated by considering all the points as belonging to the neighborhood.

The surface normal vector is one of the most important features for geometry reconstruction and interpretation of point clouds [41]. The normal vector provides information about the geometry of the underlying sampled surface. Many different applications calculate the normal vector as a local feature that is used for different purposes, including segmentation [42], surface reconstruction [43], or computer graphics [44].

2D point clouds represent a noisy sampling of curves. Information about orientation and curvature of these curves is lost in the sampling process. The estimation of the surface normal restores this information by calculating the orthogonal vector to the tangential line of the curve [45].

Considering a set of n points $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, $p_i \in \mathbb{R}^2$, where $p_i = (p_{ix}, p_{iy})^T$ represent the 2D coordinates of the measured points, the estimation of the normal vectors for p_i is $n_i = (n_{ix}, n_{iy})^T$, which are the coordinates of the perpendicular vector to the curve at that point.

The estimation of the normal vectors for point cloud data involves at least two main steps: identification of the applicable neighboring points, and estimation of the normal vector based on points in the local neighborhood.

Nearest Neighbors The normal vector is a local feature specific to a given point. Thus, reliable estimation of the normal vector largely depends on the identification of its neighboring points. The number of neighboring points, k, is critical. Too high a value for k degrades the local characteristic of the normal vector. On the contrary, too low a value for k may not be sufficient to satisfactorily represent the local geometry. An adaptive neighborhood size based on local properties, such as noise scale, curvature and sampling density can be selected [46]. However, in most applications, a fixed value of k is selected based on the expected shape of the object and the noise.

Given k, the neighborhood of point p_i, $\mathcal{Q}_i = \{q_{i1}, q_{i2}, \ldots, q_{ik}\}$, $q_{ij} \in P$, is calculated using a k-nearest neighbors algorithm. This algorithm is efficiently executed using a kd-tree.

Normal estimation One of the most efficient methods to estimate surface normals is based on the covariance matrix [47]. The eigen-analysis of the covariance matrix provides invariant descriptions of shape that indicate the orientation of the point cloud.

The first step to calculate the covariance matrix is to subtract the centroid from the point cloud, which is equivalent to translating the coordinate system to the location of the mean. This results in $\mathcal{Q}_i^z = \{q_{i1}^z = q_{i1} - \bar{q}_i, q_{i2}^z = q_{i2} - \bar{q}_i, \ldots, q_{ik}^z = q_{ik} - \bar{q}_i\}$, where $\bar{q}_i$ is calculated as (25).

$$\bar{q}_i = \frac{1}{k} \sum_{j=1}^{k} q_{ij} \tag{25}$$

The covariance matrix, $C_i \in \mathbb{R}^{2\times2}$, for $\mathcal{Q}_i$ is defined as (26).

$$C_i = \frac{1}{k} \sum_{j=1}^{k} \left(q_{ij}^z\right)\left(q_{ij}^z\right)^T \tag{26}$$

The covariance matrix is symmetrical, as seen in (27). Therefore, only the upper triangular entries (including the diagonal) must be computed. The coefficients of the matrix are calculated using (28)

$$C_i = \begin{pmatrix} a & b \\ b & d \end{pmatrix} \tag{27}$$

$$\begin{gathered} a = \tfrac{1}{k} \sum_{j=1}^{k} \left(q_{ijx}^z\right)^2,\ b = \tfrac{1}{k} \sum_{j=1}^{k} \left(q_{ijx}^z\right)\left(q_{ijy}^z\right), \\ d = \tfrac{1}{k} \sum_{j=1}^{k} \left(q_{ijy}^z\right)^2 \end{gathered} \tag{28}$$

The eigenvalues and eigenvectors of C_i determine two orthogonal vectors, one of which defines a line whose orientation minimizes, in the least square sense, the squared distance of all the points. This line is a reasonable approximation to the curve tangent. Therefore, the normal vector of $\mathcal{Q}_i$ is the eigenvector corresponding to the smallest eigenvalue of the covariance matrix C_i, that is, the principal component with the smallest covariance.

The non-zero vector v is said to be an eigenvector of matrix C_i for the eigenvalue λ when (29) is satisfied. This equation can be also written as (30), where I is the identity matrix.

$$C_i v = \lambda v \tag{29}$$

$$(C_i - \lambda I)v = 0 \tag{30}$$

The solutions of equation (30) are given by (31), which is known as the characteristic equation.

$$det(C_i - \lambda I) = 0 \tag{31}$$

The polynomial equation derived from (31) is (32), where Tr is the trace of the matrix.

$$\lambda^2 - Tr(C_i)\lambda + det(C_i) = 0 \tag{32}$$

Table 1 Summary of the registration strategies evaluated

Strategy	Iterations	Distance (mm)	Time (ms)
Naive numerical approach (model with 0.1 mm resolution)	13	0.0078	222.72
Naive numerical approach (model with 0.01 mm resolution)	19	0.0023	3299.71
Numerical approach using kd-trees (model with 0.1 mm resolution)	13	0.0078	20.22
Numerical approach using kd-trees (model with 0.01 mm resolution)	19	0.0023	38.32
Numerical approach using kd-trees (model with 0.01 mm resolution)	21	0.0011	46.39
Naive analytical approach	35	8.53E-07	323.41
Naive analytical approach with efficient estimation rigid transform	35	8.53E-07	281.86
Efficient analytical approach (transformation and closest point)	35	8.53E-07	22.58
Efficient analytical approach with R-tree	35	8.53E-07	6.76
Efficient analytical approach with primitive caching	35	8.53E-07	1.57
Efficient and robust analytical approach (caching, coarse reg., and X84 rule)	35	8.53E-07	4.83
Efficient and robust analytical approach (caching, coarse reg., and median)	35	8.53E-07	3.46
Efficient and robust analytical approach (median and subsampling 50%)	35	8.53E-07	1.70
Efficient and robust analytical approach (median and subsampling 25%)	35	8.53E-07	0.79

Table 2 Summary of the results

Model	Points	Iterations	Time (ms)
Beam	339	35	0.31
Complex shape	711	2	0.15
Real rail with outliers	458	14	0.21
Real rail with misaligned camera	505	20	0.33
Real rail with two shapes	779	23	0.50
Real rail with zoom	505	19	0.25
Real rail from the inspection system	4045	14	2.38
Real rail (50% subsampling)	4045	14	0.89
Real rail (25% subsampling)	4045	14	0.42

Substituting the coefficients of C_i in (32) and solving for λ gives the two solutions in (33). Therefore, the smallest eigenvalue, λ_{min}, is calculated as (34).

$$\lambda = \frac{(a+d) \pm \sqrt{(a-d)^2 + 4b^2}}{2} \tag{33}$$

$$\lambda_{min} = \frac{(a+d) - \sqrt{(a-d)^2 + 4b^2}}{2} \tag{34}$$

Substituting λ_{min} in (30) gives the eigenvector in (35). It can be normalized by dividing it by its magnitude. This is the normal vector of p_i considering a neighborhood of size k.

$$v = \begin{pmatrix} b \\ \lambda_{min} - a \end{pmatrix} \tag{35}$$

These simple mathematical equations can be executed very efficiently, providing a robust estimation of the normal vector of a 2D point cloud, $n_{ix} = b$ and $n_{iy} = \lambda_{min} - a$.

There is a degenerate case when a or d are equal to zero. This means the point cloud represents a vertical or horizontal line. The normal in this case is $(1,0)^T$ or $(0,1)^T$, respectively.

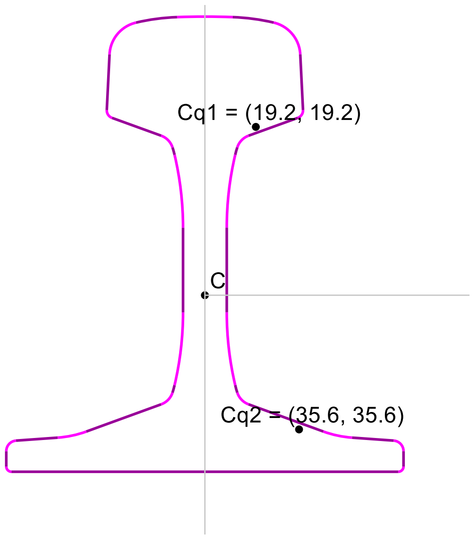


Fig. 10 Resolving the 180-degree ambiguity for a rail model

Centroid and orientation alignment The alignment of centroids is achieved by shifting the centroid of the data to the centroid of the model. The alignment of orientation is achieved by rotating the data according to the angle difference between the normal of the data and the normal of the model. This rotation creates a 180-degree ambiguity. A possible application-dependent solution to this problem is based on the analysis of the shape of the model. In the rail model used as a test, the head and the foot have a different shape. Therefore, metrics can be extracted to indicate when the data is rotated 180 degree with respect to the model. A simple and efficient approach is to compute the centroids of the data when it is divided into four quadrants. Only the centroids of the two quadrants on the right must be calculated, as can be seen in Fig. 10. The x coordinate of the centroid of the top right quadrant ($Cq1$) must be lower than the x coordinate of the centroid of the bottom right quadrant ($Cq1$). When this condition is not true for the data it indicates it is upside down, and a 180 degree rotation is required.

3.5.2 Outlier rejection

Outlier rejection is a required step for the registration process with noisy or incomplete data. Considering all correspondences as valid in these cases leads to inaccurate results. Many different strategies have been proposed [48]. In this work we consider two robust strategies that are based on statistics from the residual vector: the median threshold and the X84 rule [49].

Outlier rejection based on the median threshold is carried out by calculating the median squared distance between the potential correspondences. Only those candidates whose distance is lower than k times the squared median distance are considered valid.

Outlier rejection based on the X84 rule is a more elaborate procedure that uses robust statistics to set a rejection threshold. In this case, the threshold is obtained from the Median Absolute Deviation (MAD), which is calculated using (36), where ε represents the residuals between correspondences. The X84 rule rejects correspondences that are more than k times MAD. Assuming an underlying Gaussian distribution of the residuals, a value of $k = 5$ is usually selected, as the resulting threshold contains more than 99.9% of the distribution [50].

$$\mathrm{MAD} = \mathrm{median}_i \left(\, |\varepsilon_i - \mathrm{median}_j(\varepsilon_j)| \, \right) \tag{36}$$

The execution speed of the registration obviously increases with the addition of the initial coarse registration and the outlier rejection strategy, as more computation is required. The execution time when using the strategy based on the median is 3.46 ms. The strategy based on the X84 rule requires more calculations and is slightly slower: 4.83 ms. Tests with different types of synthetic data, noise, outliers, and real data did not provide any significant difference between any of these methods. Therefore, the fastest strategy based on the median is selected for the rejection of outliers.

More complex strategies calculate descriptive features about the correspondences to determine if they are really valid correspondences. Surface normals for all the points in the data is a common procedure. This is an effective procedure with complex shapes and where the initial coarse registration does not produce a good approximation. In this work, computing the surface normals for all the points is not considered necessary. Not only would it increase the execution time significantly, it would not provide better accuracy.

3.6 Subsampling

Subsampling consists of applying the registration with only a subset of the points in the data. Obviously, reducing the number of points will reduce the execution time. Thus, this is a simple method to reduce computation demands. However, this method is not without drawbacks. Subsampling can create aliasing, and small features of the model that are vital to determining the correct alignment may be lost. Moreover, in the presence of noise and outliers, subsampling can select those points that do not have valid correspondences in the model, producing an incorrect registration. Therefore, subsampling is a strategy that needs to be applied with care and moderation to preserve accuracy.

Among the methods to select points for subsampling, random [51] and uniform [52] subsampling are the most common strategies. More complex approaches based on surface normals provide better results but are much more computationally demanding.

Applying uniform subsampling to the test data produces the expected reduction in execution speed. Applying a sampling rate of 50%, the registration converges in 1.70 ms, and 0.79 ms with 25%. The registration includes the initial coarse registration and the outlier rejection method. The number of iterations and the accuracy obtained is the same, as the test data does not contain noise and the subsampling rate used is low.

Table 1 shows a summary of all the strategies evaluated and the performance obtained with the test data.

4 Results and discussion

In order to test the proposed registration procedure, it has been applied to different models and data, with missing parts, outliers and complex shapes. The results can be seen in Figs. 11 and 12, and Table 2.

Figs. 11a and 11b show the experiments with a beam model. In this case, the data, the point cloud that needs to be aligned, is corrupted with noise. In addition, an important part of the foot is missing. The registration aligns the shapes correctly, ignoring the missing part. The registration requires 35 iterations, and the execution time is 0.31 ms. In this case the data only contains 339 points.

Figs. 11c and 11d show the experiments with a complex shape. The data has been corrupted with Gaussian noise. The results of the registration is correct. The initial coarse registration performs a good approximation

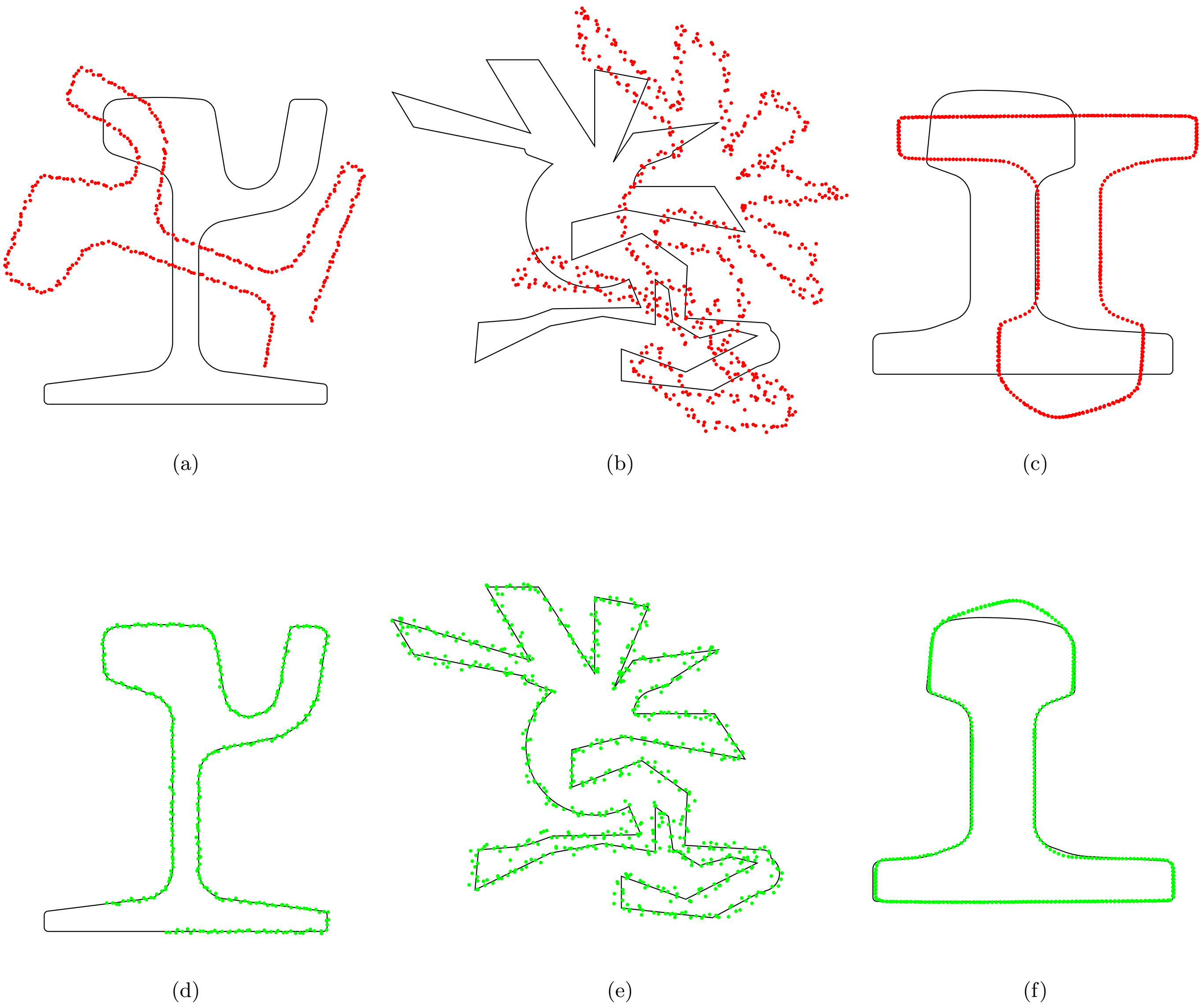


Fig. 11 Registration experiments. (a), (b), (c) Models and data. (e), (e), (f) Results of the registration.

that allows the fine registration to align the shapes perfectly. Only 2 iterations were necessary, which were executed in 0.15 ms for a point cloud with 711 points.

Figs. 11e and 11f show the experiments with a different rail model. The data was obtained with a real structured sensor but was transformed with a synthetic transformation. The objective is to test the proposed procedure with data that contains outliers, and to test the 180-degree ambiguity. Therefore, this experiment is used to test robustness. As can be seen, the registration provides the desired result. It solves the rotation ambiguity and deals with the outliers in the head correctly. This part corresponds to a defect that cannot be aligned with the model. The outlier rejection strategy worked as expected. The registration was performed in 14 iterations and took 0.21 ms for a point cloud with 458 points.

Fig. 12 shows additional tests with challenging cases. Fig. 12a shows real data with a synthetic transformation and modified to simulate a misaligned camera. This is a possible scenario that produces points in a different reference system than the points acquired by the rest of the cameras. Fig. 12c shows real data with a synthetic transformation and modified to simulate that more than one shape is visible. Fig. 12e shows another example with real data altered with a synthetic transformation and modified to simulate a zoom in the foot of the rail. This could be a possible scenario when the manufactured rail is bigger in one are than the compared CAD model. The

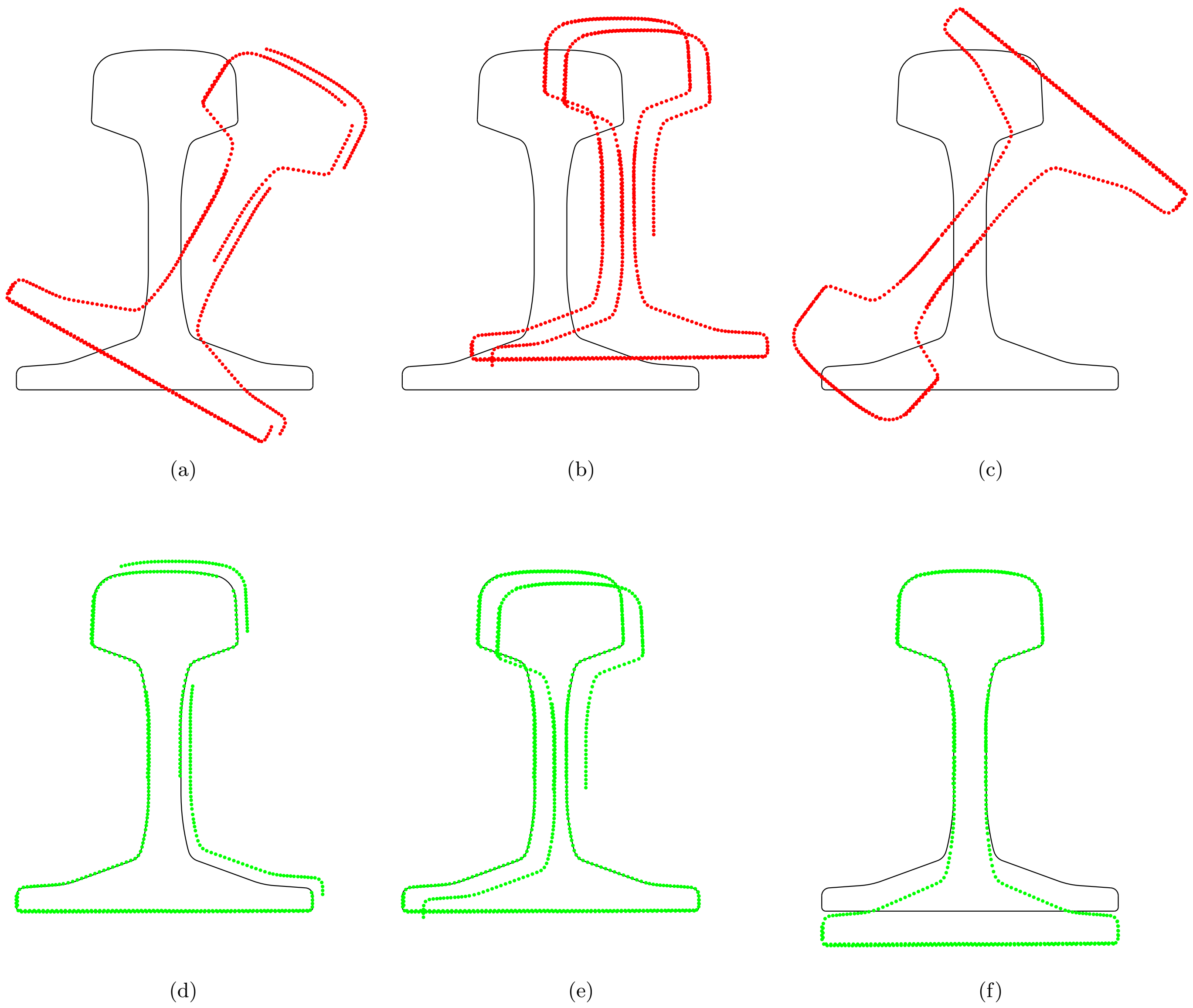


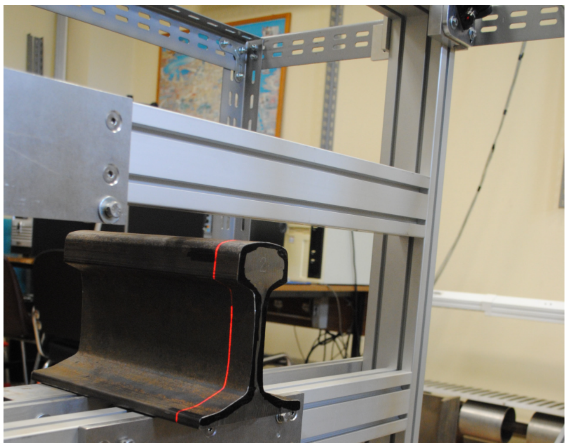
Fig. 12 Registration experiments with challenging cases. (a), (b), (c) Models and data. (d), (e), (f) Results of the registration.

registration procedure provides excellent results in all these cases. Moreover, the required execution time is in all cases below 0.6 ms, i.e., the procedure is robust and fast.

The proposed procedure has been applied to real data obtained from a rail inspection system. The system is composed of 4 laser projectors and 4 cameras with a resolution of 1400×1024 that can acquire images at 100 fps. The objective of the inspection system is to measure the dimensions of the rail and to compare these values with the reference model. Maximum allowed errors are ±0.5 mm. Therefore, the inspection system must be very accurate. Much of this accuracy comes from the registration process that aligns the model and the acquired point cloud. The acquired points that describe the surface of the rail are obtained from the images after being translated using a calibration map. However, vibrations of the rail during production require an accurate registration before metrics about the dimension of the rail are performed.

Fig. 13 shows the rail inspection system. Fig. 13a shows a laboratory prototype used to test the system, and Fig. 13b the industrial system. As can be seen, images are acquired while the rail moves, which makes the inspection of the whole rail possible.

Fig. 14 shows the images acquired by the cameras of the rail inspection system. The four cameras are re-


(a)

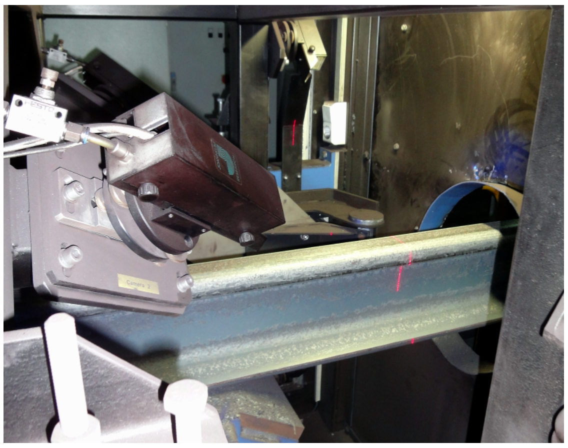

(b)

Fig. 13 Images of the rail inspection system. (a) Laboratory prototype. (b) Industrial prototype.

quired to avoid occlusion, what makes possible a correct inspection of a complete section of the rail.

Fig. 15 shows a typical example of acquired data from the rail inspection system and the registration results. Initially, data is displaced with respect to the model. The registration accurately aligns the data and the model. After registration, different metrics can be calculated about the shape, such as curvature of the arcs in the head, width at the body, or height.

Fig. 16 shows some details of different parts of the rail data. The registration aligns the data with respect to the model correctly. However, the inspected rail is rarely an exact reproduction of the model. There are some very small differences, which are generally within valid ranges. However, in some cases defects appear, which are accurately detected by the system. This process is extremely important, as using defective rails can lead to catastrophic consequences.

Registration with data acquired from the inspection system takes 2.38 ms in 14 iterations with a point cloud of 4045 points. Applying a subsampling rate of 50%, the registration converges in 0.89 ms, and 0.42 ms with 25%. Calculating the average distance between the data and the model after registration with different subsampling rates produces the results shown in Fig. 17. In this case the x axis represents the number of points used. For example, a value of 50 indicates that one point in 50 is used for registration, *i.e.*, 2%. The distance remains stable up to 20, that is, when using a subsampling rate of 5%. However, when the subsampling rate is above 12%, the number of iterations required to reach the same results increases, compensating for the reduction in execution time due to using fewer points. When using a subsampling rate of 12%, the registration converges in 0.16 ms. However, this subsampling rate means data with outliers can provoke major problems for registration. Subsampling can eliminate the required detail to produce an accurate registration. A conservative approach could be the selection of the subsampling rate based on the deadline of the registration process. For example, in an inspection system that produces 1000 fps a subsampling of 50% could be applied, or 25% for 2000 fps. These subsampling rates seem safe for the considered models in the rail inspection system. However, different models could require more points. Therefore, this decision is application-dependent.

5 Conclusions

In many real-time applications, surface registration is a crucial step for achieving high accuracy. Much research has been carried out for registration in different fields, but efficient registration for 2D has been mostly neglected. However, in different inspection systems, efficient 2D model registration is of utmost importance. Furthermore, current acquisition hardware can produce data at extremely high rates that can be used to inspect very fast moving objects. These systems require efficient algorithms for processing data. In this work, an efficient registration procedure for 2D data is proposed. Recent developments in the field are reviewed and their applicability to this problem is evaluated in terms of accuracy, speed and robustness. Moreover, efficient procedures are proposed for the most demanding parts of the registration, including the estimation of the rigid transform, the calculation of the closest points, and the estimation of surface normals. The best approach is to work with the geometric primitives that describe the model and apply these efficient procedures to estimate the correspondences. The novel primitive caching method proposed in this work further reduces the computational demands of the algorithm, which is also applicable to 3D.

Both synthetic and real data has been used to verify the performance of the proposed procedure. The re-

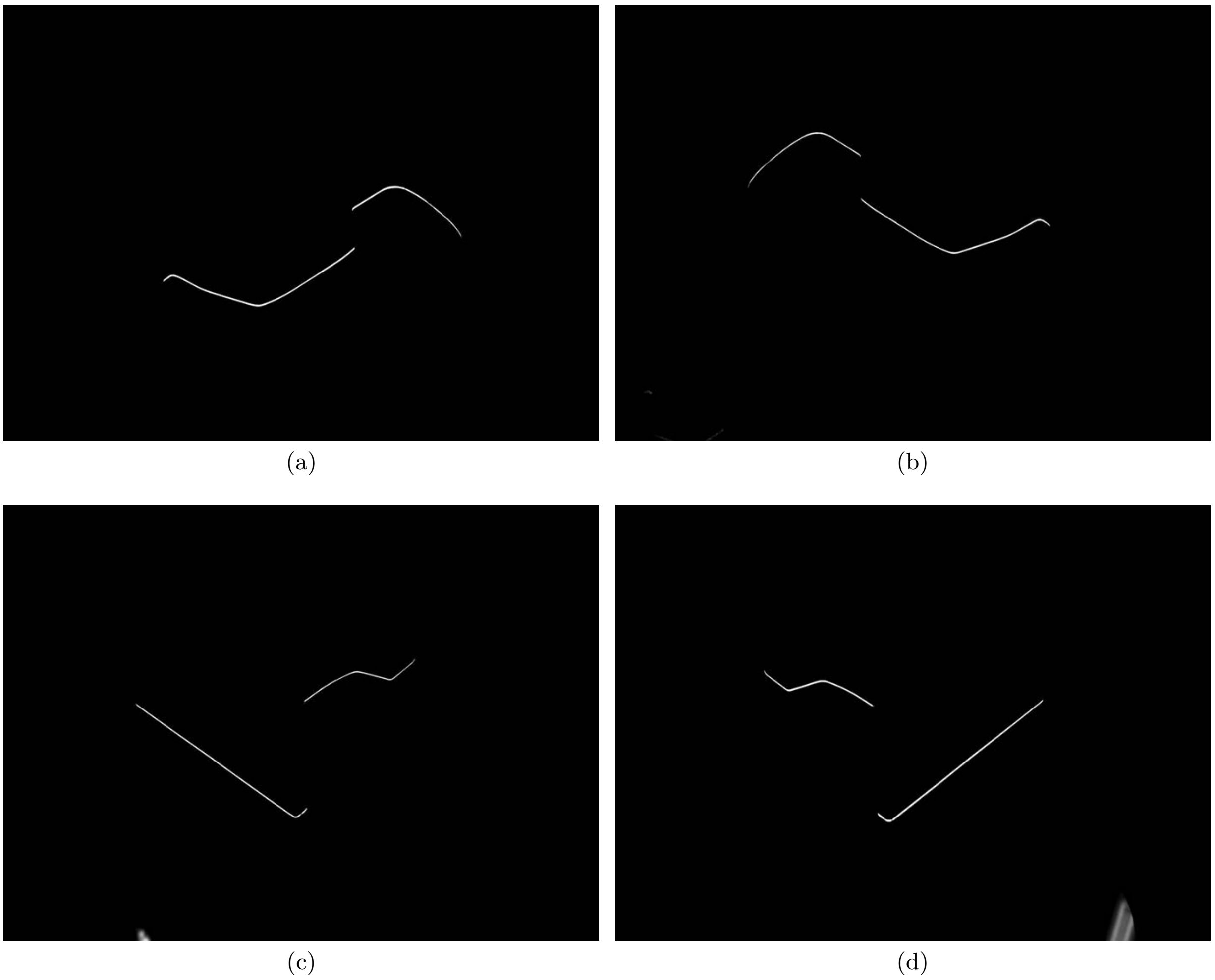

(a) (b) (c) (d)

Fig. 14 Images acquired from the projection of the laser stripes onto the rail. (a) Camera 1. (b) Camera 2. (c) Camera 3. (d) Camera 4.

sults show excellent performance. The proposed registration algorithm is a robust procedure that can very accurately align data obtained from standard structured light sensors in less than one millisecond. This provides the opportunity for the application of the registration in high-speed systems or time-constrained machine vision applications.

Acknowledgements This work has been partially funded by the project TIN2001-24903 of the Spanish National Plan for Research, Development and Innovation.

References

1. N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann, "Robust global registration," in *Symposium on geometry processing*, vol. 2, no. 3, 2005, p. 5.
2. T. Varady, R. R. Martin, and J. Cox, "Reverse engineering of geometric modelsan introduction," *Computer-Aided Design*, vol. 29, no. 4, pp. 255–268, 1997.
3. V. Raja and K. J. Fernandes, *Reverse engineering: an industrial perspective.* Springer, 2007.
4. M. Pieraccini, G. Guidi, and C. Atzeni, "3d digitizing of cultural heritage," *Journal of Cultural Heritage*, vol. 2, no. 1, pp. 63–70, 2001.
5. V. Vlahakis, N. Ioannidis, J. Karigiannis, M. Tsotros, M. Gounaris, D. Stricker, T. Gleue, P. Daehne, and L. Almeida, "Archeoguide: an augmented reality guide for archaeological sites," *IEEE Computer Graphics and Applications*, vol. 22, no. 5, pp. 52–60, 2002.
6. C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007.
7. C. Goldfeder, M. Ciocarlie, J. Peretzman, H. Dang, and P. K. Allen, "Data-driven grasping with partial sensor data," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on.* IEEE, 2009, pp. 1278–1283.
8. K. W. Bowyer, K. Chang, and P. Flynn, "A survey of approaches and challenges in 3d and multi-modal 3d+ 2d face recognition," *Computer vision and image understanding*, vol. 101, no. 1, pp. 1–15, 2006.

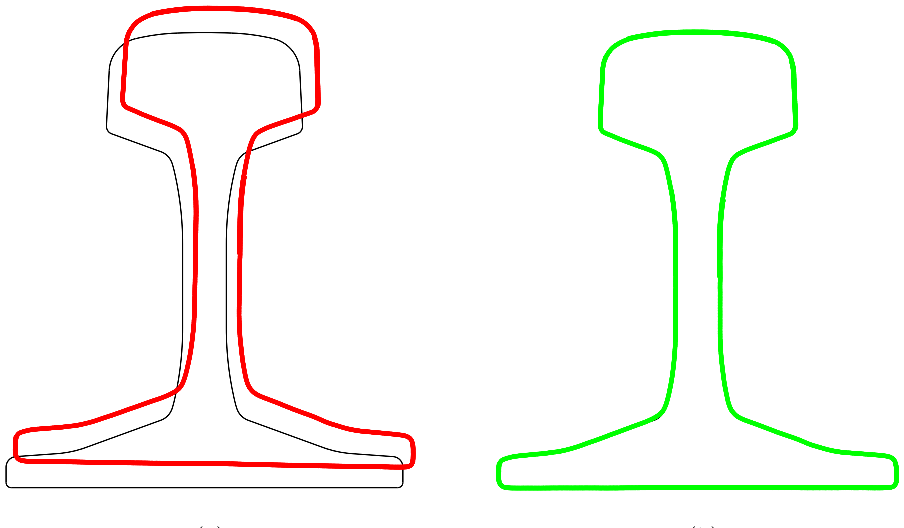

(a) (b)

Fig. 15 Rail registration with real data. (a) Rail model and data. (b) Result of the registration.

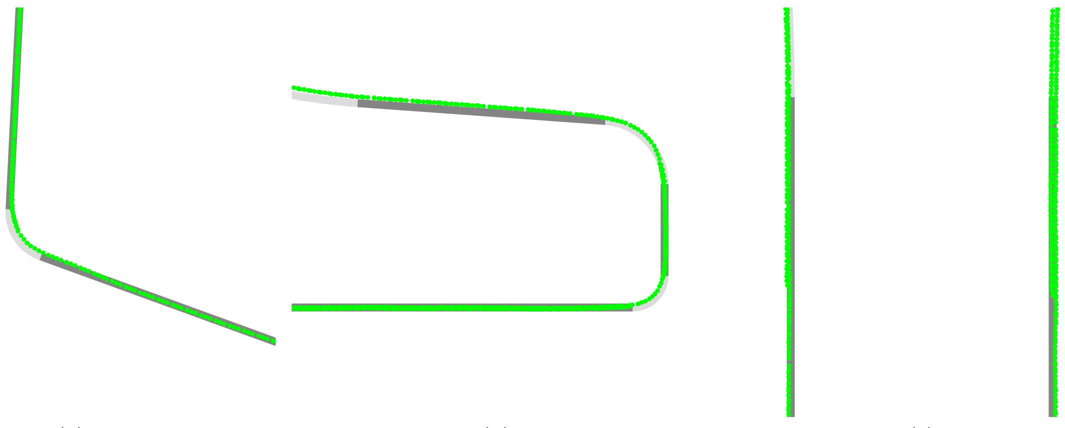

(a) (b) (c)

Fig. 16 Details of different parts of the rail data. (a) Bottom left head. (b) Right foot. (c) Body

9. U. Castellani and A. Bartoli, "3d shape registration," in *3D Imaging, Analysis and Applications*. Springer, 2012, pp. 221–264.
10. J. Salvi, J. Pages, and J. Batlle, "Pattern codification strategies in structured light systems," *Pattern Recognition*, vol. 37, no. 4, pp. 827–849, 2004.
11. C. Studholme, D. L. Hill, and D. J. Hawkes, "Automated 3-d registration of mr and ct images of the head," *Medical image analysis*, vol. 1, no. 2, pp. 163–175, 1996.
12. J. Salvi, C. Matabosch, D. Fofi, and J. Forest, "A review of recent range image registration methods with accuracy evaluation," *Image and Vision Computing*, vol. 25, no. 5, pp. 578–596, 2007.
13. P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.
14. Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.
15. H. Li, R. W. Sumner, and M. Pauly, "Global correspondence optimization for non-rigid registration of depth scans," in *Computer Graphics Forum*, vol. 27, no. 5. Wiley Online Library, 2008, pp. 1421–1430.
16. K. Pulli, "Multiview registration for large data sets," in *3-D Digital Imaging and Modeling, 1999. Proceedings. Second International Conference on*. IEEE, 1999, pp. 160–168.
17. T. Jost and H. Hugli, "A multi-resolution icp with heuristic closest point search for fast and robust 3d registration of range images," in *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*. IEEE, 2003, pp. 427–433.
18. A. W. Fitzgibbon, "Robust registration of 2d and 3d point sets," *Image and Vision Computing*, vol. 21, no. 13, pp. 1145–1153, 2003.
19. A. Rangarajan, H. Chui, E. Mjolsness, S. Pappu, L. Davachi, P. Goldman-Rakic, and J. Duncan, "A robust point-matching algorithm for autoradiograph alignment," *Medical Image Analysis*, vol. 1, no. 4, pp. 379–398, 1997.

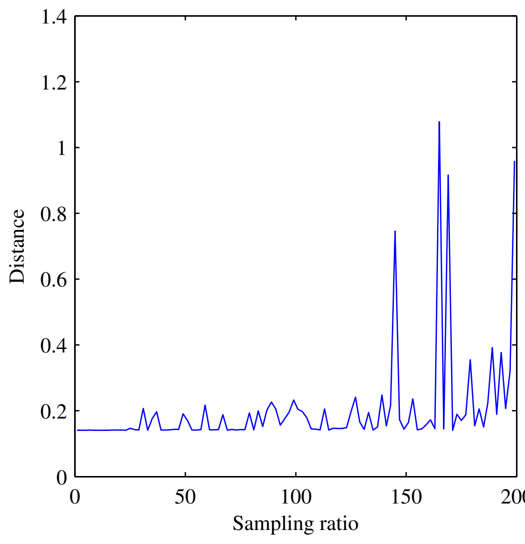


Fig. 17 Error with subsampling strategy.

20. G. C. Sharp, S. W. Lee, and D. K. Wehe, “Icp registration using invariant features,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 1, pp. 90–102, 2002.
21. S.-Y. Park and M. Subbarao, “An accurate and fast point-to-plane registration technique,” *Pattern Recognition Letters*, vol. 24, no. 16, pp. 2967–2976, 2003.
22. S. Rusinkiewicz and M. Levoy, “Efficient variants of the icp algorithm,” in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on.* IEEE, 2001, pp. 145–152.
23. A. Nuchter, K. Lingemann, and J. Hertzberg, “Cached kd tree search for icp algorithms,” in *3-D Digital Imaging and Modeling, 2007. 3DIM’07. Sixth International Conference on.* IEEE, 2007, pp. 419–426.
24. I. K. Park, M. Germann, M. D. Breitenstein, and H. Pfister, “Fast and automatic object pose estimation for range images on the gpu,” *Machine Vision and Applications*, vol. 21, no. 5, pp. 749–766, 2010.
25. M. Esteghamatian, Z. Azimifar, P. Radau, and G. Wright, “Real time cardiac image registration during respiration: a time series prediction approach,” *Journal of real-time image processing*, vol. 8, no. 2, pp. 179–191, 2013.
26. R. Usamentiaga, J. Molleda, and D. F. García, “Fast and robust laser stripe extraction for 3d reconstruction in industrial environments,” *Machine Vision and Applications*, vol. 23, no. 1, pp. 179–196, 2012.
27. N. Kehtarnavaz and M. Gamadia, “Real-time image and video processing: from research to reality,” *Synthesis Lectures on Image, Video & Multimedia Processing*, vol. 2, no. 1, pp. 1–108, 2006.
28. K. Arun, T. S. Huang, and S. D. Blostein, “Least-squares fitting of two 3-d point sets,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-9, no. 5, pp. 698–700, Sept 1987.
29. J. Elseberg, S. Magnenat, R. Siegwart, and A. Nüchter, “Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration,” *Journal of Software Engineering for Robotics*, vol. 3, no. 1, pp. 2–12, 2012.
30. J. L. Bentley, “K-d trees for semidynamic point sets,” in *Proceedings of the sixth annual symposium on Computational geometry.* ACM, 1990, pp. 187–197.
31. Y. Jia, J. Wang, G. Zeng, H. Zha, and X.-S. Hua, “Optimizing kd-trees for scalable visual descriptor indexing,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on.* IEEE, 2010, pp. 3392–3399.
32. M. Muja and D. G. Lowe, “Scalable nearest neighbor algorithms for high dimensional data,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, 2014.
33. R. Hartley and A. Zisserman, *Multiple view geometry in computer vision.* Cambridge university press, 2003.
34. D. W. Eggert, A. Lorusso, and R. B. Fisher, “Estimating 3-d rigid body transformations: a comparison of four major algorithms,” *Machine Vision and Applications*, vol. 9, no. 5-6, pp. 272–290, 1997.
35. J. C. Gower and G. B. Dijksterhuis, *Procrustes problems.* Oxford University Press Oxford, 2004, vol. 3.
36. J. H. Challis, “A procedure for determining rigid body transformation parameters,” *Journal of biomechanics*, vol. 28, no. 6, pp. 733–737, 1995.
37. P. Schneider and D. H. Eberly, *Geometric tools for computer graphics.* Morgan Kaufmann, 2002.
38. J. F. Hill, “The pleasures of perp dot products,” in *Graphics gems IV.* San Diego: Academic Press, 1994, pp. 138–148.
39. A. Guttman, *R-trees: a dynamic index structure for spatial searching.* ACM, 1984, vol. 14, no. 2.
40. D. Simon, “Fast and accurate shape-based registration,” Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1996.
41. K. Klasing, D. Althoff, D. Wollherr, and 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.
42. H. Woo, E. Kang, S. Wang, and K. H. Lee, “A new segmentation method for point cloud data,” *International Journal of Machine Tools and Manufacture*, vol. 42, no. 2, pp. 167–178, 2002.
43. H.-T. Yau, C.-C. Kuo, and C.-H. Yeh, “Extension of surface reconstruction algorithm to the global stitching and repairing of stl models,” *Computer-Aided Design*, vol. 35, no. 5, pp. 477–486, 2003.
44. M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, “Computing and rendering point set surfaces,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 9, no. 1, pp. 3–15, 2003.
45. D. OuYang and H.-Y. Feng, “On the normal vector estimation for point cloud data from smooth surfaces,” *Computer-Aided Design*, vol. 37, no. 10, pp. 1071–1079, 2005.
46. N. J. Mitra, A. Nguyen, and L. Guibas, “Estimating surface normals in noisy point cloud data,” *International Journal of Computational Geometry & Applications*, vol. 14, no. 04n05, pp. 261–276, 2004.
47. J. Berkmann and T. Caelli, “Computation of surface geometry and segmentation using covariance techniques,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 11, pp. 1114–1116, 1994.
48. J. M. Phillips, R. Liu, and C. Tomasi, “Outlier robust icp for minimizing fractional rmsd,” in *3-D Digital Imaging and Modeling, 2007. 3DIM’07. Sixth International Conference on.* IEEE, 2007, pp. 427–434.
49. U. Castellani, A. Fusiello, and V. Murino, “Registration of multiple acoustic range views for underwater scene reconstruction,” *Computer Vision and Image Understanding*, vol. 87, no. 1, pp. 78–89, 2002.
50. F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel, *Robust statistics: the approach based on influence functions.* John Wiley & Sons, 2011, vol. 114.

51. T. Masuda, K. Sakaue, and N. Yokoya, "Registration and integration of multiple range images for 3-d model construction," in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, vol. 1. IEEE, 1996, pp. 879–883.

52. G. Turk and M. Levoy, "Zippered polygon meshes from range images," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM, 1994, pp. 311–318.