Coarse Alignment for Model Fitting of Point Clouds using a Curvature-Based Descriptor

Adam Leon Kleppe, *Member, IEEE,* Lars Tingelstad, *Member, IEEE,* and Olav Egeland, *Member, IEEE*

***Abstract*—This paper presents a method for coarse alignment of point clouds by introducing a new descriptor based on the local curvature. The method is developed for model fitting a CAD model for use in robotic assembly. The method is based on selecting keypoints depending on shape factors calculated from the local covariance matrix of the surface. A descriptor is then calculated for each keypoint by fitting two spheres that describe the curvature of the surface. The spheres are calculated using conformal geometric algebra, which gives a convenient and efficient description of the geometry. The keypoint descriptors for the model and the observed point cloud are then compared to estimate the corresponding keypoints, which are used to calculate the displacement. The method is tested in several experiments. One experiment is for robotic assembly, where objects are placed on a table and their position and orientation is estimated using a 3D CAD model.**

Note to Practitioners:

***Abstract*—3D cameras can be used in robotic assembly for recognizing objects, and for determining position and orientating of parts to be assembled. In such applications 3D CAD models will be available for the objects, and point clouds representing each object can be generated for comparison with the observed point clouds from the 3D camera. It is not straightforward to use existing descriptors in this work, as the point cloud from the CAD model and the observed point cloud may differ due to different view points and potential occlusions. The method proposed in this paper is intended to be easy to apply to industrial assembly problems where there is a need for a robust estimation of the displacement of an object, either as a coarse estimate for use in grasping, or as an initial guess to use in fine registration for demanding assembly operations with close tolerances. The method exploits the curvature of the point clouds to accurately describe the surrounding surface of each point. This method serves as a basis for future industrial implementations.**

***Keywords*—keypoint descriptor, conformal geometric algebra, initial alignment, point clouds**

I. Introduction

The 3D-3D registration problem [22] is well-established in computer vision, and is still an active field of research. The problem involves two sub-problems: Calculation of the displacement between two point clouds, and estimation of the point correspondences between the point clouds [5]. A large number of methods have been proposed to solve the registration problem in 3D [32], [5], where Iterative Closest Point (ICP) [3], [6], [27] is widely used. These methods can be classified as either coarse or fine registration methods, and usually both have to be applied in order to get globally optimal solution to the registration problem. Coarse registration is typically used to find an initial alignment, which provides the initial conditions for the fine registration using, e.g., ICP.

Most of the fine registration methods, including ICP, can be described as Expectation-Maximization algorithms [22], because they alternate between solving the two sub-problems until both reach a local minimum. A known restriction with Expectation-Maximization algorithms is that they converge to locally optimal solutions. To ensure convergence to the global optimum, the algorithm either has to be expanded to include global optimization techniques, such as Go-ICP [39] or Sparse ICP [4], or it has to have good initial conditions in order to converge to the correct solution. This is achieved with coarse registration methods, such as [8], [28], [29], [31].

The coarse registration methods usually only solves one of the sub-problems: Finding the pose that aligns the two point clouds. This means that these methods do not take the point correspondences into account. Coarse registration methods can be further divided into two categories; global and local approaches [5], where the global approach tries to estimate the displacement between two point clouds using global properties such as centroid to find the translation, and global principal component analysis to find the orientation. A local approach creates a set of features or signatures in each point cloud, and search for correspondences between the features. Examples of this are Point Signatures [7], Spin Images [19] and Point Feature Histograms [28], [29].

When using local coarse registration, the focus is on creating descriptors which accurately describe a point and its surrounding surface, in such a way that a region of an object should have the same descriptor regardless of what methods were used to generate the point cloud, be it with a camera or sampled from a CAD model. This means that the descriptor has to be robust in regards to noise and to the density of the point cloud. Descriptors such as [12], [17], [19], [29], [31] achieves this robustness by discretizing the descriptor into bins. This makes it possible to filter out noise and set a known density of the point cloud and also shrink the size of the descriptor to a known size, which is important for fast computation.

As shown in [24], [18], the curvature of an object can be used to calculate different properties of the object. In [24], it is used for 2D alignment, while in [18] it is used for feature extraction.

In this paper we propose a descriptor that can be used in continuous analytic expressions, which makes it possible to formulate the correspondence problem as a continuous opti-

A. L. Kleppe, L. Tingelstad and O. Egeland are with the Department of Mechanical and Industrial Engineering, Norwegian University of Science and Technology (NTNU), NO-7491 Trondheim, Norway

This work was partially funded by the Norwegian Research Council, SFI Offshore Mechatronics, project number 237896

mization problem. The motivation is that this approach may use geometric information to a larger extent, and that this may improve the estimation of the point correspondences between the point clouds, which again provides a more accurate pose estimation.

We propose to use this type of descriptor in a new method for initial alignment of two point clouds. The method first samples the point clouds using principal component analysis at each point, then the points that are considered unique in each point cloud are labelled as keypoints, and for each of these keypoints a descriptor is generated. This descriptor is based on the fitting of two spheres, representing the curvature in two orthogonal directions of the surface. The keypoint descriptor is calculated using conformal geometric algebra. The descriptors of both point clouds are then compared to estimate the point correspondence between the keypoints using least-squares optimization. The displacement that aligns the descriptors of two point clouds is then found, resulting in an initial alignment between the two point clouds.

This paper provides an extensive amount of experiments that validates the method. This contributed to a modification to the method presented in [20], which is has a more stable point correspondence estimation on point clouds with planar surfaces.

This paper is organized as follows: Section II is the preliminaries, which introduces the parts of conformal geometric algebra used in the paper. Section III describes the proposed method. Section IV shows the conducted experiment, where a 3D camera captures a point cloud of a table, extracts the point cloud of the desired object and uses the proposed method to find the initial alignment between the captured point cloud and a 3D model, as well as performing a fine estimation algorithm on it. The proposed method is also compared with other state-of-the-art descriptors. The result from these experiments are then presented and discussed, and lastly the conclusion is found in Section VI.

II. Preliminaries

A. 3D-3D Registration Problem

Consider the point cloud $\boldsymbol{X} = \{\boldsymbol{x}_i\}$, $i = 1, \dots, n_x$ of observations $\boldsymbol{x}_i \in \mathbb{R}^3$, and the point cloud $\boldsymbol{Y} = \{\boldsymbol{y}_j\}$, $j = 1, \dots, n_y$ of model point positions $\boldsymbol{y}_i \in \mathbb{R}^3$, where the model points are assumed to be calculated from a CAD model of an object. The 3D-3D registration problem is then to minimize the error function

$$E(\boldsymbol{R}, \boldsymbol{t}) = \sum_{i=1}^{n_x} ||\boldsymbol{y}_{j^*} - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}||^2 \tag{1}$$

with respect to $\boldsymbol{R}$ and $\boldsymbol{t}$, and $\boldsymbol{y}_{j^*}$ is the model point corresponding to the data point $\boldsymbol{x}_i$. In the ICP method, the model point $\boldsymbol{y}_{j^*}$ corresponding to the data point $\boldsymbol{x}_i$ is found from

$$j^* = \underset{j}{\operatorname{argmin}} ||\boldsymbol{y}_j - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}|| \tag{2}$$

The solution is then found by iteration where at each step the correspondence is found from the minimization of (2) for the current estimate of the pose, and then the estimate of the pose $\boldsymbol{R}, \boldsymbol{t}$ is found by minimizing (1) for the current estimate of the correspondence. This minimization will require that the initial guess for the pose and the correspondence is sufficiently close to the optimal solutions.

Initial alignment can be performed with a local approach using descriptors for points in the model and observation point clouds, and then to find the initial pose by matching the two point clouds based on these descriptors. Such descriptors can be calculated for all points in the point cloud, as in [28], or for keypoints that are selected based on some criterion.

B. Conformal Geometric Algebra

We will use methods based on Conformal Geometric Algebra [11], [16] in this paper as this is a formulation that is well suited to do calculations on points, planes and spheres, which will be used to describe descriptors for the point clouds, and to do optimization based on these descriptors. The Conformal Geometric Algebra extends the Euclidean space $\mathbb{R}^3$ with basis vectors given by the orthogonal unit vectors e_1, e_2, e_3 to the 5 dimensional space $\mathbb{R}^{4,1} = \text{span}\{e_1, e_2, e_3, e_0, e_\infty\}$ where $e_i \cdot e_j = \delta_{ij}$ for $i, j \in \{1, 2, 3\}$ where δ_{ij} is the Kronecker delta, $e_0 \cdot e_0 = e_\infty \cdot e_\infty = 0$ and $e_0 \cdot e_\infty = -1$.

Consider a Euclidean point $\boldsymbol{p} = p_1 e_1 + p_2 e_2 + p_3 e_3 \in \mathbb{R}^3$, which can be given by the column vector $[\boldsymbol{p}] = [p_1, p_2, p_3]^{\mathrm{T}}$. This point can be represented by the conformal point $\boldsymbol{P} \in \mathbb{R}^{4,1}$ defined by

$$\boldsymbol{P} = \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^2 e_\infty + e_0 \tag{3}$$

where $\boldsymbol{P}$ has the property that $\boldsymbol{P}_1 \cdot \boldsymbol{P}_2 = -\frac{1}{2}||\boldsymbol{p}_1 - \boldsymbol{p}_2||$. The conformal point $\boldsymbol{P}$ can also be written as the column vector

$$[\boldsymbol{P}] = \begin{bmatrix} [\boldsymbol{p}] \\ \frac{1}{2}\boldsymbol{p}^2 \\ 1 \end{bmatrix} \tag{4}$$

A plane can be given by

$$\boldsymbol{\Pi} = \boldsymbol{n} + \delta e_\infty \tag{5}$$

where $\boldsymbol{n}$ is the unit normal of the plane, while δ is the distance from the origin to the plane. This is referred to as a dual plane in [11]. The vector representation of $\boldsymbol{\Pi}$ is

$$[\boldsymbol{\Pi}] = \begin{bmatrix} [\boldsymbol{n}] \\ \delta \\ 0 \end{bmatrix} \tag{6}$$

A sphere $\boldsymbol{S}$ is denoted as

$$\boldsymbol{S} = \boldsymbol{P}_C - \frac{1}{2}r^2 e_\infty \tag{7}$$

where $\boldsymbol{P}_C$ is the center point of the sphere and r is the radius of the sphere. The vector representation of $\boldsymbol{S}$ is

$$[\boldsymbol{S}] = \begin{bmatrix} [\boldsymbol{p}] \\ \frac{1}{2}(\boldsymbol{p}^2 - r^2) \\ 1 \end{bmatrix} \tag{8}$$

The radius r of a given sphere $\boldsymbol{S}$ could be found using

$$\begin{aligned}\boldsymbol{S}\cdot\boldsymbol{S} &= (\boldsymbol{P}_C - \frac{1}{2}r^2e_\infty)\cdot(\boldsymbol{P}_C - \frac{1}{2}r^2e_\infty) \\ &= \boldsymbol{P}_C\cdot\boldsymbol{P}_C - r^2e_\infty\cdot\boldsymbol{P}_C + \frac{1}{4}r^4e_\infty\cdot e_\infty \\ &= r^2\end{aligned} \tag{9}$$

and the distance d from the center of the sphere $\boldsymbol{S}$ to a point $\boldsymbol{P}$ can be found by

$$\begin{aligned}d^2 &= \boldsymbol{S}\cdot\boldsymbol{S} - 2\boldsymbol{S}\cdot\boldsymbol{P} \\ &= r^2 - 2(\boldsymbol{p}_C + \frac{1}{2}(\boldsymbol{p}_C^2 - r^2)e_\infty + e_0)\cdot(\boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^2e_\infty + e_0) \\ &= r^2 - 2\boldsymbol{p}\cdot\boldsymbol{p}_C + (\boldsymbol{p}_C^2 - r^2) + \boldsymbol{p}^2 \\ &= \boldsymbol{p}^2 - 2\boldsymbol{p}\cdot\boldsymbol{p}_C + \boldsymbol{p}_C^2 \\ &= (\boldsymbol{p} - \boldsymbol{p}_C)^2\end{aligned} \tag{10}$$

III. Method

A. Introduction

The method, which can be described as a coarse registration method, is based on the following steps: First keypoints are selected in the model point cloud Y and observation point cloud X based on the geometry of the neighbourhood of each point. Then a descriptor is calculated for each keypoint in Y and X . Then a point correspondence is established between the keypoints in Y and X using the descriptors. Finally, the pose is estimated using the point correspondence from the keypoint matching.

B. Selection of keypoints

1) Covariance matrix: To solve the correspondence problem, a few points, called keypoints, are selected to represent each point cloud, which are used to find the same or equivalent points in the second point cloud. This has two effects: One is that it reduces the number of points in the computations, which increases the execution speed. The second is that it is more likely to find the correct correspondences when the search space is reduced in this way.

For each point $\boldsymbol{p}_i$ a neighbourhood $\boldsymbol{N}_i$ is defined as the set of all points in a ball of radius r about $\boldsymbol{p}_i$. A covariance matrix $\mathbf{C}_{\boldsymbol{p}_i}$ is calculated for all the points in this neighbourhood according to

$$\mathbf{C}_{\boldsymbol{p}_i} = \sum_{\boldsymbol{p}_k\in\boldsymbol{N}_i}([\boldsymbol{p}_k] - \bar{\boldsymbol{p}}_i)([\boldsymbol{p}_k] - \bar{\boldsymbol{p}}_i)^{\mathrm{T}} \tag{11}$$

where $\bar{\boldsymbol{p}}_i = \frac{1}{n}\sum_{\boldsymbol{p}_k\in\boldsymbol{N}_i}[\boldsymbol{p}_k]$, and n is the number of points in $\boldsymbol{N}_i$. The eigenvalues of $\mathbf{C}_{\boldsymbol{p}_i}$ are denoted λ_i, and eigenvectors are $\boldsymbol{v}_i$ for $i = 1, 2, 3$.

2) Shape factors: Keypoints are the points where the neighbouring points represents a unique shape of the point cloud. To determine which points to choose, we first have to analyze the shape around each point. This is done using the eigenvalues and eigenvectors calculated earlier.

An ellipsoid

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \tag{12}$$

can be formed around the center $\bar{\mathbf{p}}_i$ with the eigenvectors $\boldsymbol{v}_{\boldsymbol{p}_i}$ as the principal axes and the eigenvalues $\lambda_{\boldsymbol{p}_i}$ and $a = \lambda_1$, $b = \lambda_2$, $c = \lambda_3$, where $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ and $\lambda_{\boldsymbol{p}_i} = [\lambda_1\ \lambda_2\ \lambda_3]^{\mathrm{T}}$

With this ellipsoid, we can evaluate the shape of the surrounding points. If $\lambda_1 > 0, \lambda_2 = \lambda_3 = 0$, then the surrounding points are on a line in the direction of the eigenvector $\mathbf{v}_1$, while if $\lambda_1 = \lambda_2 > 0, \lambda_3 = 0$, then all the points lie on the plane spanned from the eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$. If $\lambda_1 = \lambda_2 = \lambda_3$, then the surrounding points form a sphere or an otherwise voluminous form.

Knowing this, we can classify the surface of the point cloud at the specific point, by using three shape factors [1], [26]

$$C_l = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3} \tag{13}$$

$$C_p = \frac{2(\lambda_2 - \lambda_3)}{\lambda_1 + \lambda_2 + \lambda_3} \tag{14}$$

$$C_s = \frac{3\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \tag{15}$$

where $C_l + C_p + C_s = 1$, which means that the three shape factors are less than or equal to unity.

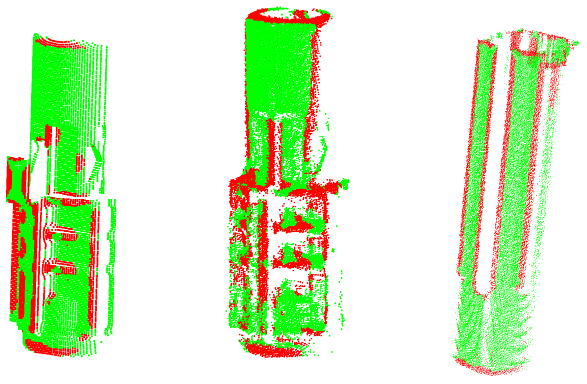
The neighbourhood of a point can then be classified as follows: If $C_l = 1$, the neighbourhood forms a linear shape; if $C_p = 1$, the neighbourhood has a planar form; and if $C_s = 1$ the neighbourhood has a spherical form.

3) Selecting keypoints: We define a point $\boldsymbol{p}_i$ to be a keypoint if the neighbourhood $\boldsymbol{N}_i$ of the point has at least $n_{\min}$ points, and the shape factors satisfy the conditions

$$C_l \geq \delta_l \text{ or } C_p \geq \delta_p \text{ or } C_s \geq \delta_s \tag{16}$$

where $n_{\min}$, δ_l, δ_p and δ_s are user-specified keypoint parameters. The parameter n_{i} defines the minimum of neighbouring points that is required. This parameter is used to ensure that the calculated shape factors for $\boldsymbol{p}_i$ are reliable. With too few neighbouring points, which happens with outliers, each point will have a big impact on the eigenvalues, and therefore the shape factors, which means that measurement noise has a large impact on the shape factors. By having a large n_{i}, we can effectively filter out these outliers.

When considering the values of these parameters, one has to take into account the shape of the point cloud and the density of points. The goal is to use these parameters to select as few points from both point clouds, but also that the points are chosen from the same regions on the point clouds. For instance, a point cloud with many large flat surfaces benefits from having $\delta_p > 1$, effectively disregarding the C_p shape factor, and choosing a high δ_l and δ_s, in a range from $0.3-0.5$, which will select the points that are not in the flat surface regions, but rather the edged surfaces. An example of point selection is shown in Fig 1.



(a) A CAD model view of object A, with the approximately same view angle as in Fig 1b.
(b) A 3D camera measurement of object A
(c) A 3D camera measurement of object B

Fig. 1. A sample of the keypoint selection process using (16) with the parameters $n = 200$, $\delta_l = 0.3$, $\delta_p = 1$ and $\delta_s = 0.3$. It is seen that the keypoints in Fig 1a and Fig 1b are similar, while that of Fig 1c is different. This is a wanted behaviour as the match between the point cloud in Fig 1a and Fig 1b will be better than that of Fig 1a and Fig 1c.

4) Automatic generation of keypoint parameters: When comparing large sets of point clouds it is tedious to manually select keypoint parameters $n_{\min}$, δ_l, δ_p and δ_s. It would then be beneficial to analyze each point cloud and automatically determine which points seem more unique than others, and generate keypoint parameters based on this.

We define following function to rank the points in a point cloud

$$F(\boldsymbol{p}_i) = \begin{cases} C_{\mathrm{l}}C_{\mathrm{p}} + C_{\mathrm{p}}C_{\mathrm{s}} + C_{\mathrm{s}}C_{\mathrm{l}}, & |\boldsymbol{N}_i| \geq n_{\min} \\ \frac{1}{3}, & |\boldsymbol{N}_i| < n_{\min} \end{cases} \tag{17}$$

where $|\boldsymbol{N}_i|$ is the number of points in the neighbourhood $\boldsymbol{N}_i$, and C_l, C_p and C_s are calculated for the point $\boldsymbol{p}$ using (13), (14) and (15) respectively. Since $C_l + C_p + C_s = 1$, $F = 0$ if either $C_l = 1$, $C_p = 1$ or $C_s = 1$, which represents a very unique point. $F = \frac{1}{3}$ means that $C_l = C_p = C_s = \frac{1}{3}$, which is not a unique point, which is the maximum value of F.

By arranging each point $\boldsymbol{p}_i$ in a point cloud from the lowest to the highest value of $F(\boldsymbol{p}_i)$, we can select k points with the lowest score which, will be the keypoints selected from the point cloud. In other words,

$$\boldsymbol{X}_{\text{keypoints}} = \{\boldsymbol{p}_i : i = 1, \ldots, k, k < |\boldsymbol{X}|\} \tag{18}$$

where k is a user-defined parameter. In order to sample the same type of points from $\boldsymbol{Y}$ we need to estimate the parameters that would yield the same results as in $\boldsymbol{X}_{\text{keypoints}}$.

To do this we perform an algorithm

```
$\delta_l = \delta_p = \delta_s = 1$
for all $\boldsymbol{p} \in \boldsymbol{X}_{\text{keypoints}}$ do
    if $C_l = \max(C_l, C_p, C_s)$ and $C_l < \delta_l$ then
        $\delta_l = C_l$
    else if $C_p = \max(C_l, C_p, C_s)$ and $C_p < \delta_p$ then
        $\delta_p = C_p$
    else if $C_s = \max(C_l, C_p, C_s)$ and $C_s < \delta_s$ then
        $\delta_s = C_s$
    end if
end for
```

where $\boldsymbol{X}_{\text{keypoints}}$ are the k points in $\boldsymbol{X}$ with the lowest S. The algorithm effectively groups the keypoints into three groups, one where C_l is the maximum, one where C_p is the maximum and one where C_s is the maximum. The algorithm then checks each group and selects the corresponding δ to be the lowest value of C within each group.

This gives an estimate of δ_l, δ_p and δ_s which can be used to pick the keypoints $\boldsymbol{Y}_{\text{keypoints}}$ in $\boldsymbol{Y}$ using (16). The $n_{\min}$ parameter is still dependent on the point density of $\boldsymbol{Y}$, and cannot be estimated from the keypoints in $\boldsymbol{X}_{\text{keypoints}}$, however, unless there is a significant difference in the point density between $\boldsymbol{X}$ and $\boldsymbol{Y}$ then $n_{\min}$ can be chosen to be the same for both $\boldsymbol{X}$ and $\boldsymbol{Y}$.

C. Generation of keypoint descriptors

In order to compare keypoints from $\boldsymbol{X}_{\text{keypoints}}$ and $\boldsymbol{Y}_{\text{keypoints}}$, we need to find a measurable comparison between them. This is done by generating a descriptor for each keypoint which describes the shape of the keypoint and can be used to compare with other descriptors.

To do this, we define the curvature along the surface where each keypoint lie. This is done by generating two spheres, one which describes the curvature along the least curving direction of the surface as given by the eigenvector $\boldsymbol{v}_1$ of the covariance matrix $\mathbf{C}_{\boldsymbol{p}_i}$, and the orthogonal direction as given by the eigenvector $\boldsymbol{v}_2$.

Note that the covariance matrix was computed from (11) at an earlier step of the method, and that this covariance matrix defines an ellipsoid which is fitted to the points of the neighbourhood in the sense that the point of the neighbourhood forms the volume of the ellipsoid. In contrast to this, the neighbourhood is regarded as the surface of the spheres that are fitted in this step, which means that these spheres have a different geometry from the ellipsoid defined by the covariance matrix.

To estimate these spheres, we use the method for n-sphere fitting to a set of points using Conformal Geometric Algebra [10]. The motivation for this is that conformal geometric algebra provides a convenient and very efficient description of spheres, and the distance between points and spheres. Note that the algorithms of the implementation can be formulated efficiently in terms of linear algebra. The method reduces to a to a Pratt fit [25] in the case that the sphere fit is reduced to a circle fit, as pointed out in [10].

The method generates a 5×5 covariance matrix for a set of points $\boldsymbol{P}_i$

$$\mathbf{C} = \sum_{i=0}^{n}([\boldsymbol{P}_i][\boldsymbol{P}_i]^{\mathrm{T}})\mathbf{G} \tag{19}$$

where $\boldsymbol{P}_i \in \boldsymbol{X}_{\text{keypoints}}$ are conformal points and

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix} \tag{20}$$

In order to find the sphere $\boldsymbol{S}$ we can find the eigenvector corresponding to the smallest eigenvalue, $\mathbf{v}_5$, and define it as the dual of the optimal sphere estimate

$$[\boldsymbol{S}] = \mathbf{v}_5 \tag{21}$$

In order to generate the sphere that represents the least curvature, a weight is added to each point when calculating $\mathbf{C}$. First we define the plane $\boldsymbol{\Pi}_{\text{kp}_1}$ which is spanned from $\boldsymbol{v}_1$ and $\boldsymbol{v}_3$ and also intersects $\boldsymbol{P}_{\text{kp}} \in \boldsymbol{X}_{\text{keypoints}}$, where $\boldsymbol{v}_1$ and $\boldsymbol{v}_3$ are the eigenvectors found using the covariance matrix describing the neighbourhood of $\boldsymbol{P}_{\text{kp}}$, $\boldsymbol{N}[\text{kp}]$.

Since $\boldsymbol{v}_1$ is the eigenvector that corresponds to the eigenvalue λ_1, it also represents the direction of most variance in regards to the neighbourhood $\boldsymbol{N}_{\text{kp}}$. The direction with the most variance, when considering surfaces, is also the direction with the least curvature. The plane $\boldsymbol{\Pi}_1$ is therefore the plane which cuts the surface along the least curved direction.

By extending the n-sphere method to a weighted sum equation

$$\mathbf{C}_{\boldsymbol{P}_{\text{kp}_1}} = \sum_{i=0}^{n}([\boldsymbol{P}_i][\boldsymbol{P}_i]^{\mathrm{T}} e^{-w|\boldsymbol{P}_i \cdot \boldsymbol{\Pi}_{\text{kp}_1}|})\mathbf{G} \tag{22}$$

where $\boldsymbol{P}_i \in \boldsymbol{X}_{\text{keypoints}}$, w is a weight parameter, and $|\boldsymbol{P}_i \cdot \boldsymbol{\Pi}_{\text{kp}_1}|$ is the distance from the point $\boldsymbol{p}_i$ to the plane $\boldsymbol{\Pi}_{\text{kp}_1}$. The weight is calculated so that all points that lie on the plane $\boldsymbol{\Pi}_{\text{kp}_1}$ are given the weight of 1, and the weight will decrease the further the point is from the plane. This makes the points close to $\boldsymbol{\Pi}_{\text{kp}_1}$, i.e. the points that represents the curvature along the least curving direction are weighted higher than the ones further away.

The weight element can be viewed as a Gaussian distribution

$$f(x, \mu, \sigma) = \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{23}$$

where $\boldsymbol{P}_i \cdot \boldsymbol{\Pi}_{\text{kp}_1} = -(x-\mu)^2$ and $w = \frac{1}{2\sigma^2}$.

We can then use the covariance matrix $\mathbf{C}_{\text{kp}}$ to find the sphere $\boldsymbol{S}_{\text{kp}_1}$ by finding the eigenvector which corresponds to the smallest eigenvalue, i.e. $\mathbf{v}_5$.

$$[\boldsymbol{S}_{\text{kp}_1}] = \mathbf{v}_5 \tag{24}$$

To find the curvature along the direction orthogonal to that of $\mathbf{v}_1$, we calculate yet another plane, $\boldsymbol{\Pi}_{\text{kp}_2}$, which is spanned from $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$ and intersects the same point $\boldsymbol{P}_{\text{kp}} \in \boldsymbol{X}_{\text{keypoints}}$, where $\boldsymbol{v}_2$ and $\boldsymbol{v}_3$ are the eigenvectors are found using the covariance matrix describing the neighbourhood of $\boldsymbol{P}_{\text{kp}}$, $\boldsymbol{N}[\text{kp}]$.

This is again used to calculate the covariance matrix

$$\mathbf{C}_{\boldsymbol{P}_{\text{kp}_2}} = \sum_{i=0}^{n}([\boldsymbol{P}_i][\boldsymbol{P}_i]^{\mathrm{T}} e^{-w|\boldsymbol{P}_i \cdot \boldsymbol{\Pi}_{\text{kp}_2}|})\mathbf{G} \tag{25}$$

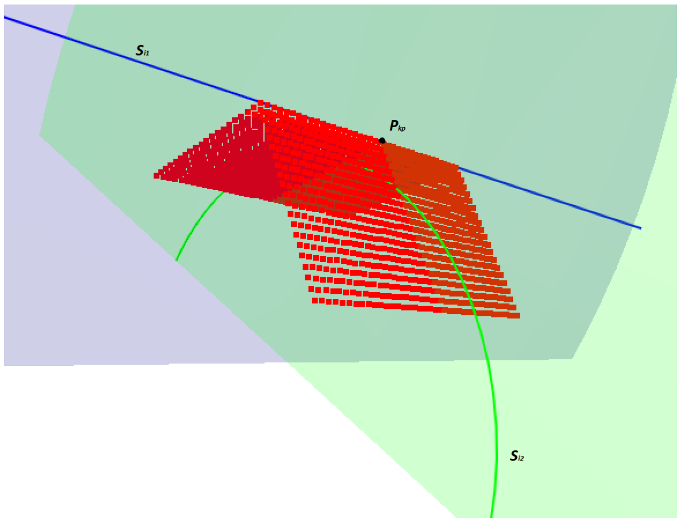


Fig. 2. An example of a descriptor. The spheres are shown as circles in the figure to make it easier to view. The blue circle of $\boldsymbol{S}_1$ lies on the blue $\mathbf{v}_1$-$\mathbf{v}_3$ plane, while the green circle of $\boldsymbol{S}_2$ lies on the green $\mathbf{v}_2$-$\mathbf{v}_3$ plane. Note that the green sphere does not intersect with the keypoint $\boldsymbol{p}_{\text{kp}}$.

which describes the curvature along the direction orthogonal to that of $\mathbf{v}_1$. We can then generate the sphere $\boldsymbol{S}_{\text{kp}_2}$ by using the eigenvector corresponding to the smallest eigenvalue of $\mathbf{C}$.

$$[\boldsymbol{S}_{\text{kp}_2}] = \mathbf{v}_5 \tag{26}$$

With these two spheres we can define the descriptor for the keypoint $\boldsymbol{p}_{\text{kp}}$

$$\boldsymbol{F}_{\text{kp}} = \{\boldsymbol{S}_{\text{kp}_1}, \boldsymbol{S}_{\text{kp}_2}\} \tag{27}$$

An example of such one descriptor can be seen in Fig 2. In the figure, sphere fitting cases of Fig 4 are used in the two orthogonal planes $\mathbf{v}_1$-$\mathbf{v}_3$ and $\mathbf{v}_2$-$\mathbf{v}_3$. This generates a descriptor which is unique for each keypoint and can accurately describe the surrounding surface. It can be seen that the blue sphere has an almost infinite radius, which is because in that direction the point cloud is flat as a plane. If the point cloud was a corner, then both the green and blue spheres would be equal and with a small radius.

Fig 3 shows a set of examples of different surfaces, and the estimated spheres that are calculated. The different values of r and d can be seen in Fig 4.

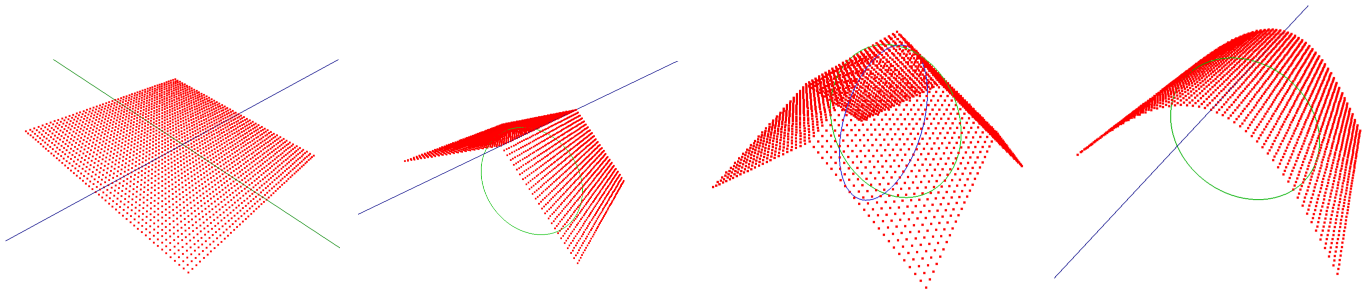
This process is then repeated on all keypoints in $\boldsymbol{X}_{\text{keypoints}}$ and $\boldsymbol{Y}_{\text{keypoints}}$.

D. Estimating point correspondences

When all the descriptors are generated, it is possible to estimate the correspondences between $\boldsymbol{X}_{\text{keypoints}}$ and $\boldsymbol{Y}_{\text{keypoints}}$.

To find the corresponding point to $\boldsymbol{p}_k \in \boldsymbol{X}_{\text{keypoints}}$ we solve the equation

$$\min \epsilon(\boldsymbol{p}_k, \boldsymbol{p}_l), \quad \forall\ \boldsymbol{p}_l \in \boldsymbol{Y}_{\text{keypoints}} \tag{28}$$



(a) A flat shape. $\boldsymbol{S}_1$ (blue) and $\boldsymbol{S}_2$ (green) are both flat and therefore has $d \approx r \approx \infty$.

(b) A wedge shape. $\boldsymbol{S}_1$ (blue) has the parameters $r \approx d \approx \infty$. $\boldsymbol{S}_2$ (green) is a edge shape, resulting in $d > r$.

(c) A corner shape. $\boldsymbol{S}_1$ (blue) and $\boldsymbol{S}_2$ (green) are both edge shapes and therefore have $d > r$.

(d) A curved shape. $\boldsymbol{S}_1$ (blue) has the parameters $r \approx d \approx \infty$. $\boldsymbol{S}_2$ (green) follows the curve along the surface, resulting in $r \approx d$.

Fig. 3. A sample of different types of surfaces, and their corresponding r and d values. The sphere estimates are marked as a blue circle for $\boldsymbol{S}_1$ and a green circle for $\boldsymbol{S}_2$. The spheres are represented by circles to simplify the figure.

where

$$\epsilon(\boldsymbol{p}_k, \boldsymbol{p}_l) = (r_{k1} - r_{l1})^2 + (d_{k1} - d_{l1})^2 \tag{29}$$
$$+ (r_{k2} - r_{l2})^2 + (d_{k2} - d_{l2})^2 \tag{30}$$

and

$$\begin{aligned} r_{k1}^2 &= \boldsymbol{S}_{k1} \cdot \boldsymbol{S}_{k1} \\ d_{k1}^2 &= \boldsymbol{S}_{k1} \cdot \boldsymbol{S}_{k1} - 2\boldsymbol{S}_{k1} \cdot \boldsymbol{P}_k \\ r_{k2}^2 &= \boldsymbol{S}_{k2} \cdot \boldsymbol{S}_{k2} \\ d_{k2}^2 &= \boldsymbol{S}_{k2} \cdot \boldsymbol{S}_{k2} - 2\boldsymbol{S}_{k2} \cdot \boldsymbol{P}_k \end{aligned} \tag{31}$$

$$\begin{aligned} r_{l1}^2 &= \boldsymbol{S}_{l1} \cdot \boldsymbol{S}_{l1} \\ d_{l1}^2 &= \boldsymbol{S}_{l1} \cdot \boldsymbol{S}_{l1} - 2\boldsymbol{S}_{l1} \cdot \boldsymbol{P}_l \\ r_{l2}^2 &= \boldsymbol{S}_{l2} \cdot \boldsymbol{S}_{l2} \\ d_{l2}^2 &= \boldsymbol{S}_{l2} \cdot \boldsymbol{S}_{l2} - 2\boldsymbol{S}_{l2} \cdot \boldsymbol{P}_l \end{aligned} \tag{32}$$

where r_{k1}, r_{k2}, r_{l1} and r_{l2} are the radii of $\boldsymbol{S}_{k1}$, $\boldsymbol{S}_{k2}$, $\boldsymbol{S}_{l1}$ and $\boldsymbol{S}_{l2}$ respectively, and d_{k1}, d_{k2}, d_{l1} and d_{l2} are the distances between the center of $\boldsymbol{S}_{k1}$ and $\boldsymbol{p}_k$, $\boldsymbol{S}_{k2}$ and $\boldsymbol{p}_k$, $\boldsymbol{S}_{l1}$ and $\boldsymbol{p}_l$ and $\boldsymbol{S}_{l2}$ and $\boldsymbol{p}_l$ respectively. The results from different sphere estimates and the relationship between r and d can be seen in Fig 4.

E. Pose estimation

At this stage, all the points $\mathbf{x}_i$ in $\boldsymbol{X}_{\text{keypoints}}$ has an estimated correspondence to a point $\mathbf{y}_j$ in $\boldsymbol{Y}_{\text{keypoints}}$. With this correspondence, the pose can be found by minimizing (1) for the point clouds $\boldsymbol{X}_{\text{keypoints}}$ and $\boldsymbol{Y}_{\text{keypoints}}$ defined by the keypoints. This is straightforward, and can be done in the usual way using Singular Value Decomposition (SVD) as described in, e.g., [2], [37].

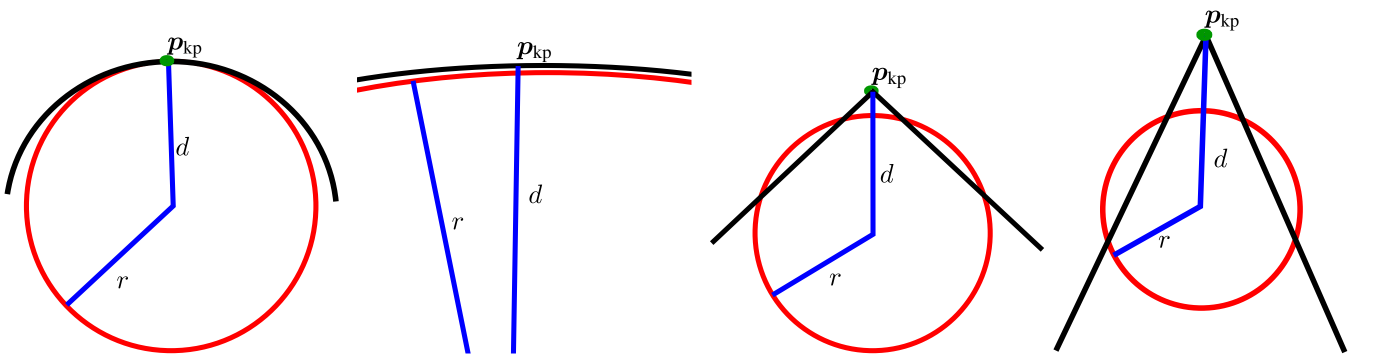
In this work, the pose estimation was done in terms of Conformal Geometric Algebra using the method of [38]. The reason for this was that the estimation of the spheres of the descriptors was based on geometric algebra, and it was decided to use this also for the pose estimation. It turned out the pose estimation gave as good as identical accuracy with the methods of [38] and the SVD method of [2], [37], and that the SVD method had 10 % less computation time.

To proceed, it is necessary to introduce the outer product [11], [16]. The outer product of two basis vectors in $\mathbb{R}^{4,1}$ satisfies $e_i \wedge e_j = -e_j \wedge e_i$, $i, j \in \{1, 2, 3, \infty, 0\}$, and it follows that $e_i \wedge e_i = 0$. The outer product of basis vectors is written $e_i \wedge e_j = e_{ij}$, which is of grade 2, $e_i \wedge e_j \wedge e_k = e_{ijk}$ which is of grade 3, and so on. Note that a repeated index means that the outer product is zero, which follows from $e_{ii} = 0$. The highest grade nonzero outer product of basis vectors is $e_{0123\infty}$, which is of grade 5, and is called the pseudoscalar. The geometric product of two basis vectors is written $e_i e_j = e_i \cdot e_j + e_i \wedge e_j$. Calculation rules for geometric products of more than two basis elements are found in [11].

The conformal geometric algebra over the space $\mathbb{R}^{4,1}$ is $\mathbb{G}_{4,1} = \text{span}\{1, e_1, e_2, e_3, e_0, e_\infty, e_{23}, e_{31}, e_{12}, \ldots, e_{0123\infty}\}$, which is closed under the geometric product UV of two elements $U, V \in \mathbb{G}_{4,1}$. It is noted that the basis elements of $\mathbb{G}_{4,1}$ are of grade 0, 1, 2, 3, 4, and 5. An element of $\mathbb{G}_{4,1}$ is called a multivector, and is given by $U = \sum_I u_I e_I$ where $u_I \in \mathbb{R}$ are scalar coordinates, and I denotes the indices of the basis elements of $\mathbb{G}_{4,1}$. The reverse of a multivector is given by $\widetilde{U} = \sum_I u_I \widetilde{e}_I \in \mathbb{G}_{4,1}$, where $\widetilde{e}_I$ means that the ordering of the factors in each basis element has been reversed, e.g., $\widetilde{e}_{ij} = e_{ji}$ and $\widetilde{e}_{ijk} = e_{kji}$. Then the geometric product of two multivectors $U = \sum_I u_I e_I$ and $V = \sum_J v_J e_J$ is given by $UV = \sum_I \sum_J u_I v_J e_I e_J$. The scalar part of the geometric product UV is denoted by $\langle UV \rangle$.

The pose can described in terms of a screw displacement defined by a rotation θ about a line, and a translation d along the same line. Let the line be given in Plücker coordinates by the direction vector $\boldsymbol{a} = a_1 e_1 + a_2 e_2 + a_3 e_3$ and the moment $\boldsymbol{b} = b_1 e_1 + b_2 e_2 + b_3 e_3$, where $\boldsymbol{a} \cdot \boldsymbol{b} = 0$. Then in conformal geometric algebra the pose can be described by the motor [35]

$$\begin{aligned} M =& \cos\frac{\theta}{2} + \sin\frac{\theta}{2} A \\ &+ \varepsilon\left(\frac{d}{2}\cos\frac{\theta}{2} A + \sin\frac{\theta}{2} B - \frac{d}{2}\sin\frac{\theta}{2}\right) \end{aligned} \tag{33}$$



(a) Sphere estimate of a spherical surface. $r \approx d$. (b) Sphere estimate of a planar surface. $r \approx d \approx \infty$. (c) Sphere estimate of an edge. $r < d$. (d) Sphere estimate of a point. $r \ll d$.

Fig. 4. A sample of resulting sphere estimates on different surfaces. The different values of r and d indicate what the different shapes are.

where $A = a_1e_{23} + a_2e_{31} + a_3e_{12}$, $B = b_1e_{23} + b_2e_{31} + b_3e_{12}$ and $\varepsilon = e_{321\infty}$ is the dual unit. From this it can be seen that a motor M is in the 8 dimensional linear space

$$\mathbb{M} = \mathrm{span}\{1, e_{23}, e_{31}, e_{12}, e_{1\infty}, e_{2\infty}, e_{3\infty}, e_{123\infty}\} \tag{34}$$

Let c_i, $i = 1, \dots, 8$ denote the basis elements of $\mathbb{M}$, that is, $c_1 = 1, c_2 = e_{23}, c_3 = e_{31}, \dots$. Then the motor can be written $M = \sum_{i=1}^{8} m_i c_i$. It is seen that the motor M can be described in terms of the coordinate vector $\boldsymbol{m} = [m_1, \dots, m_8]^{\mathrm{T}}$, which is partitioned as $\boldsymbol{m} = [\boldsymbol{r}^{\mathrm{T}}, \boldsymbol{t}^{\mathrm{T}}]^{\mathrm{T}}$. Here $\boldsymbol{r}$ and $\boldsymbol{t}$ are four-dimensional coordinate vectors, where $\boldsymbol{r}$ describes the rotation and $\boldsymbol{t}$ describes the translation of the displacement described by M. It can be shown that the motor M given by (33) satisfies $\widetilde{M}M = 1$. Therefore M is a motor if and only if

$$M \in \mathbb{M} \quad \text{and} \quad \widetilde{M}M = 1 \tag{35}$$

The error function ϵ_k for between the point $\mathbf{x}_k \in \boldsymbol{X}_{\text{keypoints}}$ and the corresponding point $\mathbf{y}_k \in \mathbf{Y}_{\text{keypoints}}$, is defined as

$$\epsilon_k = -\frac{1}{2}d_k^2 = (\widetilde{M}\mathbf{y}_k M) \cdot \mathbf{x}_k = \langle \widetilde{M}\mathbf{y}_k M \mathbf{x}_k \rangle \tag{36}$$

The pose estimation problem can then be formulated as

$$\max_M \sum_{k=1}^{n} \langle \widetilde{M}\mathbf{y}_k M \mathbf{x}_k \rangle, \quad \widetilde{M}M = 1 \tag{37}$$

where n is the number of points in $\boldsymbol{X}_{\text{keypoints}}$. To solve this optimization problem, we use the method in [38]. The operator $\mathcal{L}$ is defined by $\mathcal{L}M = \sum_{k=1}^{n} \mathbf{y}_k M \mathbf{x}_k$, so that the optimization problem can be written

$$\max_M \langle \widetilde{M}\mathcal{L}M \rangle, \quad \widetilde{M}M = 1 \tag{38}$$

It is noted that

$$\langle \widetilde{M}\mathcal{L}M \rangle = \sum_{i=1}^{8}\sum_{j=1}^{8} m_i m_j \langle \widetilde{e}_i \mathcal{L} e_j \rangle = \boldsymbol{m}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{m} \tag{39}$$

where $\boldsymbol{Q} = \{Q_{ij}\}$, and $Q_{ij} = \langle \widetilde{e}_i \mathcal{L} e_j \rangle$.

Let the subspace $\overline{\mathbb{M}}$ be given by

$$\overline{\mathbb{M}} = \mathrm{span}\{1, \widetilde{e}_{23}, \widetilde{e}_{31}, \widetilde{e}_{12}, e_{10}, e_{20}, e_{30}, e_{3210}\} \tag{40}$$

Let the basis elements of $\overline{\mathbb{M}}$ be denoted c^i, $i = 1, \dots, 8$. Then the basis elements c^i of $\overline{\mathbb{M}}$ will be reciprocal to the basis elements c_i of $\mathbb{M}$, which means that $c^j \cdot c_i = \langle c^j c_i \rangle = \delta_{ij}$ and δ_{ij} is the Kronecker delta. The projection of a multivector $Y \in \mathbb{G}_{4,1}$ onto $\overline{\mathbb{M}}$ is given by $\boldsymbol{P}_{\overline{M}}(Y) = \sum_{i=1}^{8} c^i \langle c_i Y \rangle$.

Define the matrix $\boldsymbol{L} = \{L_{ij}\}$ by

$$L_{ij} = \sum_{i=1}^{k} \langle \widetilde{e}_i \boldsymbol{P}_{\overline{M}}(\mathcal{L}e_j) \rangle \tag{41}$$

and let $\boldsymbol{L}$ be partitioned into 4×4 submatrices, such that

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_{rr} & \mathbf{L}_{rt} \\ \mathbf{L}_{tr} & \mathbf{L}_{tt} \end{bmatrix} \tag{42}$$

Then the 4×4 matrix $\mathbf{L}'$ is defined by

$$\mathbf{L}' = \mathbf{L}_{rr} - \mathbf{L}_{rt}(\mathbf{L}_{tt}^{+}\mathbf{L}_{tr}) \tag{43}$$

where $\mathbf{L}_{tt}^{+}$ denotes the Moore-Penrose pseudoinverse. The coefficient vector $\boldsymbol{r}$ of the rotation can then be found as the eigenvector of $\mathbf{L}'$ associated with the largest eigenvalue. This gives the rotation with the smallest rotation angle. The coefficient vector $\boldsymbol{t}$ of the translation can be found by computing

$$\boldsymbol{t} = -(\mathbf{L}_{tt}^{+}\mathbf{L}_{tr})\boldsymbol{r} \tag{44}$$

Then the motor M is given by the coordinate vector $\boldsymbol{m} = [\boldsymbol{r}^{\mathrm{T}}, \boldsymbol{t}^{\mathrm{T}}]^{\mathrm{T}}$.

IV. Experiments

The proposed method was compared with a selection of state-of-the-art methods for initial alignment. These methods were Fast Point Feature Histograms (FPFH) [28], Point-Pair Features (PPF) [12], Signature of Histogram of OrienTation (SHOT) [36], 3D Shape Context (3DSC) [13] and Globally Aligned Spatial Distribution (GADS) [23].

There were a total of three experiments conducted. The first was where two instances of the same point cloud had a known

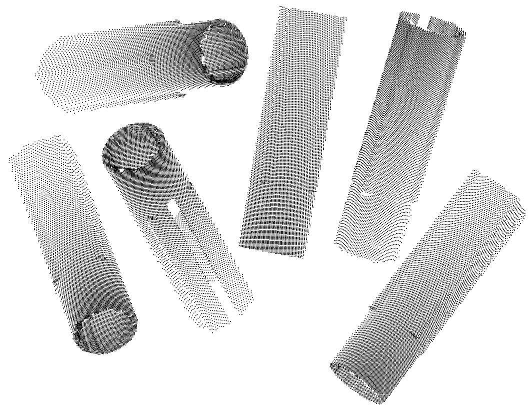


Fig. 5. Multiple point clouds of the same 3D CAD model, from multiple views

displacement between each other, and the proposed method was run several times with different parameters, in order to analyze what impact each parameter had. In the second experiment, two instances of the same point clouds had a known displacement between them and one was subjected to different Gaussian noise. Both the proposed method, FPFH, PPF, SHOT, 3DSC amd GASD were used, and their performance was evaluated. In the last experiment, the position of a 3D model in a scene was estimated. Here, each method tried to find the alignment between two different point clouds, one generated from a 3D model and the other captured by a 3D camera, where the displacement was not known. This demonstrates a real world application where one tries to estimate the position of an object in a scene with only the use of a 3D model.

A. Setup

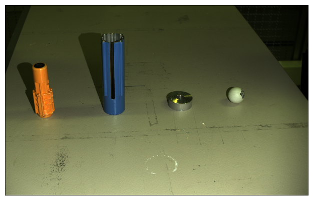
1) Hardware: The hardware that was used for the experiments, was the same for all three. The computer that was used was a desktop computer with an Intel Core i7 7700k Sky Lake at $4.2\,\mathrm{GHz}$ with 32GB $2666\,\mathrm{MHz}$ DDR4 and a EVGA GeForce GTX 1080 Founders Edition graphics card. The computer was running Ubuntu 16.04 LTS.

The point clouds that were taken with a 3D camera, were taken using the Zivid 3D camera provided by ZividLabs [33]. The Zivid 3D camera outputs 2.3 Mpixel RGBD image, with a field of view of $425{\times}267\,\mathrm{mm}$ at a distance of $0.6\,\mathrm{m}$ with a depth resolution of $0.1\,\mathrm{mm}$ at the same distance.

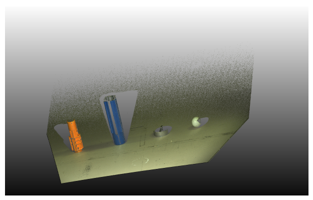
2) Point Cloud Data: There were a total of 129 point clouds used in the experiments.

84 of these were generated from two 3D CAD models, where a simulated 3D camera generated point clouds from 42 different angles around each CAD model. A sample of which is shown in Fig 5.

10 of the point clouds were taken with the Zivid camera. The camera captured a total of 4 scenes where a set of objects were placed on a table. Each object was detected using the object detection method described in [34], where the proposed RANSAC method was used to find the table, and the region growing algorithm was used to separate the objects on the table from each other. The method for object recognition proposed in the paper was not used, and the different objects were separated manually. The object detection method cannot handle occluded parts. Each object was saved as an individual point cloud, resulting in 10 point clouds. A sample is shown in Fig 6.

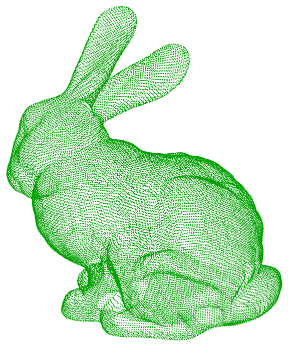


(a) 2D image taken with the Zivid Camera

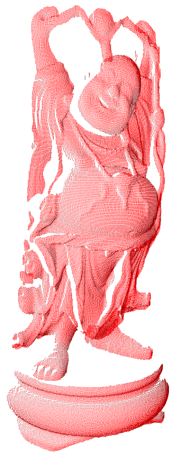


(b) Point cloud acquired by the Zivid Camera

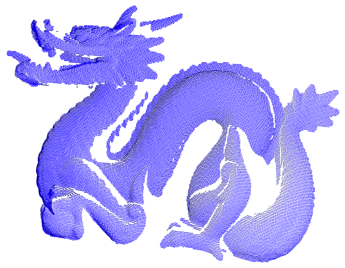
Fig. 6. The objects on the table, where the orange and blue objects are detected and their pose are estimated. The image is taken with the Zivid Camera.



(a) 3D point cloud of the Stanford Bunny.



(b) 3D point cloud of the Stanford Happy Buddha



(c) 3D point cloud of the Stanford Dragon

Fig. 7. A sample of the database point clouds used.

35 point clouds were from the Stanford database [21], where 5 of them were of the Stanford Bunny, 15 of the Stanford Happy Buddha and 15 of the Stanford Dragon. Fig 7 shows an example of these point clouds.

3) Implementation: The proposed method was implemented using the versor library [9] together with the Eigen library [15].

The parameters that were used in the method were selected using the results from the first experiments. The r parameter in (11) was $6.3\,\mathrm{mm}$ and w in (22) was 0.2. The algorithm performed by estimating C_l(13), C_p(14), and C_s(15) as described in Section III-B4, where n in (16) was 200 and the number of selected keypoints were 2000 which was approximately between $5\,\%$ and $10\,\%$ of the total point cloud for the whole data set.

FPFH, PPF, SHOT and 3DSC were implemented using the PCL library [30], and was implemented using the sample codes that were provided on their websites, or other supporting websites. The parameters were chosen to be similar to the proposed method to the extent it was possible. The point clouds were first down-sampled using a voxel grid of $1\,\mathrm{mm}$, followed by the normal estimation algorithm in Section II with a radius of $30\,\mathrm{mm}$. FPFH, PPF and SHOT had a search radius of $30\,\mathrm{mm}$ when using the KD-tree, and SHOT had in addition the radius of a plane defined as $1000\,\mathrm{mm}$. 3DSC had a slight variation in the parameter selection, because if they were chosen in the same manner as the rest, it failed. It had a normal estimation radius and search radius of the KD-tree set to $40\,\mathrm{mm}$ and a minimum radius for the search sphere set to $4\,\mathrm{mm}$ and a point per radius density parameter set to 8. After each method had generated a descriptor, a RANSAC algorithm was performed with 1000 iterations and an inlier threshold of $5\,\mathrm{mm}$. GADS was partly implemented using the PCL library, while the rest was implemented in C++. The original implementation in [23] uses color data to generate the global descriptor. Since the CAD models do not have color data, it was not possible to perform this step of the descriptor generation. The GASD descriptor used in this paper only generates a descriptor based on the shape of the point clouds.

All code was implemented using C++11, and was compiled using O3 optimization. There was no parallelism involved in the implementation.

B. *Experiment 1*

1) Description: The first experiment took two instances of the same point cloud with a known displacement between them. The proposed method then performed a pose estimation between the two point clouds with various parameters. Each test was performed with changing the radius r in (11), the weight w in (22) and the number of keypoints as described in (16). When one parameter was changed, the others stayed constant. This was performed on all 129 point clouds.

Since the two point clouds were the same, and no noise was involved, both the exact displacement and the point correspondences were known. This made it possible to know the error in displacement and the error in the estimated point correspondence.

The radius r was tested at $0.5\,\mathrm{mm}$, $0.8\,\mathrm{mm}$, $1\,\mathrm{mm}$, $2\,\mathrm{mm}$, $5\,\mathrm{mm}$, $10\,\mathrm{mm}$ and $20\,\mathrm{mm}$. The weight w was tested at 10, 5, 2, 1, 0.8, 0.5, 0.2, 0.1, 0.05, 0.01 and 0.001. The sample size was tested at 20000, 10000, 5000, 2000, 1000, 800, 500 and 100. The n_min was 200.

2) Results: The results from the three tests that were performed are shown in Fig 8, Fig 9 and Fig 10. It is seen in the figures that there is a correlation between the accuracy of the method and the r and w parameters. The figures show 8 chosen point clouds; one for each of the two objects taken with the Zivid camera, two from generated point clouds from the 3D model from each of the two objects, a total of four, and two point clouds from the Standford Bunny set. The point clouds were chosen randomly.

The best fit for the radius r is between $5\,\mathrm{mm}$ and $10\,\mathrm{mm}$ when it comes to correspondence depending on the point cloud, while the pose estimate stays approximately the same. As expected, the execution time grows exponentially depending on the radius, which is due to the fact that the amount of points used in the Principal Components Analysis (PCA) method increases exponentially with the radius.

The weight w had little to no impact on the translation estimate, but a higher weight value makes a more unstable rotation estimate. The point correspondence the optimal solution ranges between 0.05 and 0.2 depending on the point cloud. It can be seen that there is a correlation between the stability of the rotation estimate and the point correspondence. The weight had little impact on the execution time.

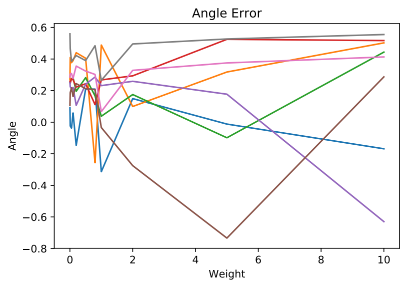
The number of keypoints had little to no impact on the method's performance. The pose estimation stayed approximately the same, while the number of corresponding points were linear, which is expected. This meant that about $99\,\%$ of the points were accurately estimated. The point clouds where the graph caps off at a maximum value, is where the point cloud uses all the points in the point cloud, and can therefore not select more keypoints. The execution time escalates, which is expected given that there are more keypoints to work through.

This experiment indicates that a careful selection of the r and w parameters is needed to ensure optimal results and performance, while the number of keypoints has little impact on the method's accuracy. The experiment also indicates that the method performs equally well in regards to pose estimation.

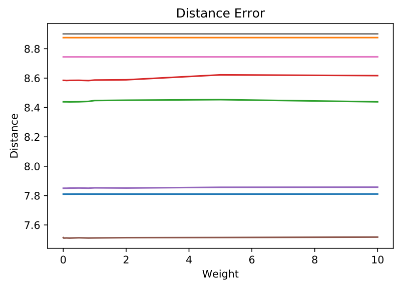
C. *Experiment 2*

1) Description: In the second experiment, the pose between two of the same point cloud was estimated, where one point cloud was displaced with a known displacement, and each point within that point cloud had added a Gaussian noise. The added noise had a μ of $3\,\mathrm{mm}$ and σ at 0.1, 0.3, 0.5, 1.0, 2.0. This was tested on all 129 point clouds with both the proposed method, FPFH, PPF, SHOT, 3DSC and GADS. The accuracy of the point correspondences were not evaluated, since the indexing of the points in the point clouds are mixed up when using the PCL library. Only the pose estimation error was evaluated.

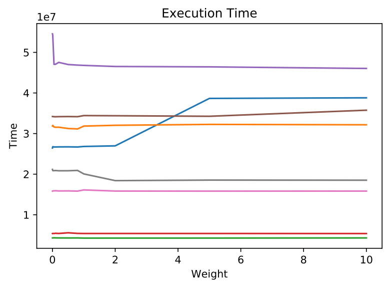
2) Results: The results from the experiment are shown in Table I and Table II The results shows that the fastest method is by far FPFH, and also that it is the least accurate, which is not a desired result for industrial applications. Both SHOT and the proposed method perform equally fast, however, the proposed method performs with better accuracy. The only method that



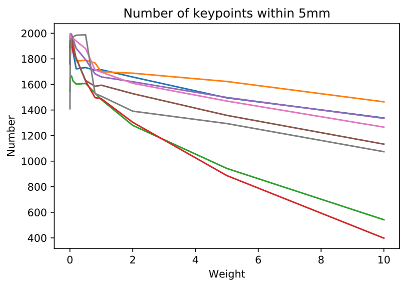
(a) Angle error [rad], measured using an angle-axis calculation of the error transformation



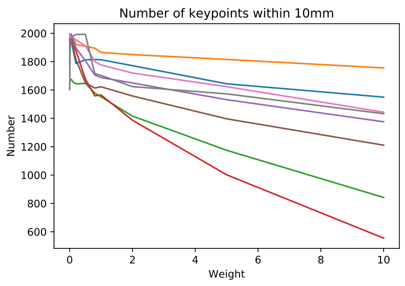
(b) Distance error [mm], measured using the distance of the error transformation



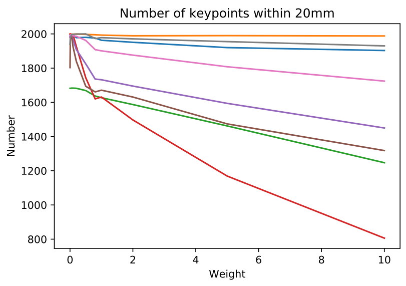
(c) Execution time [μs]



(d) Number of times where the estimated correspondence was within 5 mm away from the correct corresponding point



(e) Number of times where the estimated correspondence was within 10 mm away from the correct corresponding point



(f) Number of times where the estimated correspondence was within 20 mm away from the correct corresponding point

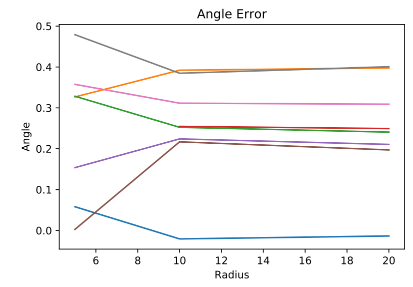
Fig. 8. Results when adjusting the weight parameter. Each coloured graph represents the results from a sample point cloud, where 8 are selected from the total of 129 point clouds. The red, green and orange graphs are of different viewpoint point clouds of part A, the blue, purple and pink graphs are of different viewpoint point clouds of part B, and the brown and gray graphs are two point clouds of the Stanford Bunny.

TABLE I. RESULTS FROM EXPERIMENT 2. THE RESULTS SHOW THE AVERAGE DISTANCE AND ANGLE ERROR OVER ALL 129 POINT CLOUDS FOR EACH NOISE INTERVAL.

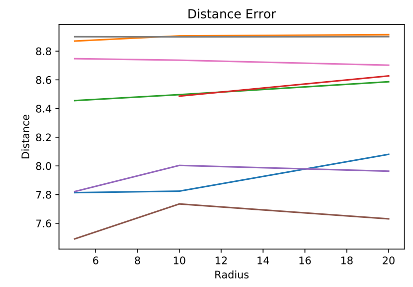
	Noise $\sigma = 0.1$		Noise $\sigma = 0.3$		Noise $\sigma = 0.5$		Noise $\sigma = 1.0$		Noise $\sigma = 2.0$	
	Angle [rad]	Distance [mm]	Angle [rad]	Distance [mm]	Angle [rad]	Distance [mm]	Angle [rad]	Distance [mm]	Angle [rad]	Distance [mm]
Proposed	0.122	0.645	0.132	0.833	0.151	0.944	0.262	1.267	0.643	3.562
FPFH	0.231	101.436	0.271	112.312	0.431	163.647	0.642	287.961	0.851	1663.624
PPF	0.095	1.451	0.114	1.729	0.272	2.534	0.296	3.833	0.577	6.996
SHOT	0.340	6.282	0.349	6.544	0.537	8.964	0.699	10.070	0.798	11.947
3DSC	0.242	16.125	0.365	17.938	0.777	22.153	0.825	23.775	0.846	31.858
GADS	0.542	5.341	0.601	7.625	0.677	10.511	0.751	11.753	0.910	11.884

TABLE II. RESULTS FROM EXPERIMENT 2. THE RESULTS SHOW THE AVERAGE EXECUTION TIME OVER ALL 129 POINT CLOUDS FOR EACH NOISE INTERVAL.

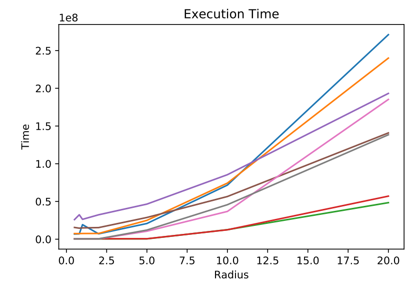
	Noise $\sigma = 0.1$	Noise $\sigma = 0.3$	Noise $\sigma = 0.5$	Noise $\sigma = 1.0$	Noise $\sigma = 2.0$
	Execution Time [ms]	Execution Time [ms]	Execution Time [ms]	Execution Time [ms]	Execution Time [ms]
Proposed	2329.517	2328.523	2330.665	2430.101	2354.512
FPFH	829.594	843.941	901.421	830.512	854.442
PPF	17721.757	17883.121	17519.337	17329.881	19901.731
SHOT	2026.415	2139.533	2101.112	2323.121	1997.454
3DSC	16507.965	16433.103	17031.315	16831.610	17124.155
GADS	1402.442	1421.531	1411.595	1399.931	1420.40



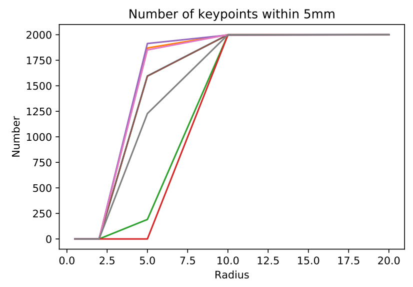
(a) Angle error [rad], measured using an angle-axis calculation of the error transformation



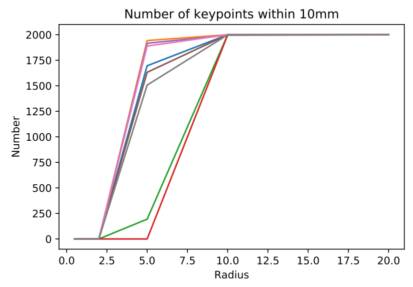
(b) Distance error [mm], measured using the distance of the error transformation



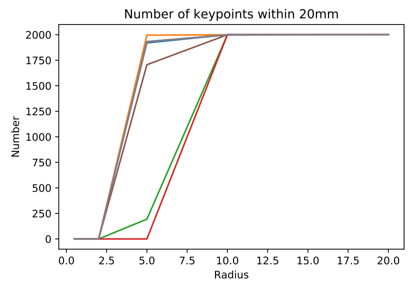
(c) Execution time [µs]



(d) Number of times where the estimated correspondence was within 5 mm away from the correct corresponding point



(e) Number of times where the estimated correspondence was within 10 mm away from the correct corresponding point



(f) Number of times where the estimated correspondence was within 20 mm away from the correct corresponding point

Fig. 9. Results when adjusting the radius parameter. Each coloured graph represents the results from a sample point cloud, where 8 are selected from the total of 129 point clouds. The red, green and orange graphs are of different viewpoint point clouds of part A, the blue, purple and pink graphs are of different viewpoint point clouds of part B, and the brown and gray graphs are two point clouds of the Stanford Bunny.

performs as accurate as the proposed method is PPF, but it is the slowest of all the methods. It is worth noting that though the mean error is slightly high on the overall results, the methods had some cases with very accurate results. The PPF method for instance got accurate results on point clouds that had little features, such as the Stanford Bunny, where the other methods had significantly larger errors. In this case the proposed method got a angle error of 0.847 radians.

D. Experiment 3

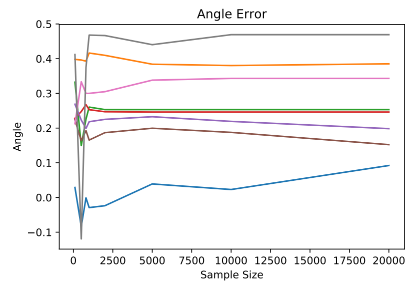
1) Descriptions: The last experiment was a real world demo, where a Zivid camera captured a 3D image of a table with a set of objects on it, see Section IV-A2. Each of the 10 point clouds of the real world objects were compared to a selection of the 84 generated point clouds of the CAD models, and the best fit was estimated as well as the pose between the CAD model and the scene. This effectively estimated the position of the object relative to the camera.

Since there are no known point correspondence nor known displacements, an ICP algorithm was performed after every estimation. This was done using the CloudCompare software [14], which provided the final transformation as well as an root mean square calculation of the estimated point correspondences. This together with a visual inspection was sufficient to evaluate the performance of each method.

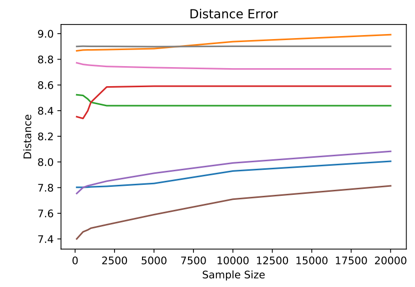
TABLE III. RESULTS FROM EXPERIMENT 3. THE RMS SHOWS THE RMS BETWEEN THE CORRESPONDING POINTS BEFORE APPLYING ICP BETWEEN THE TWO COMPARED POINT CLOUDS. THE COMMENT DESCRIBES SOME OF THE REMARKS THAT WERE DONE WITH THE VISUAL INSPECTION OF THE RESULTS.

Method	RMS [mm]	Execution time [ms]	Comment
Proposed	2.70914	21487.476	
FPFH	5.94532	2390.431	Point cloud was flipped upside down
PPF	90.01690	56926.563	Estimate before ICP was far away from the actual point cloud
SHOT	3.35757	30411.992	Got a better RMS with a different point cloud
3DSC	3.04315	163224.032	
GADS	4.85295	1224.032	The orientation was not correct

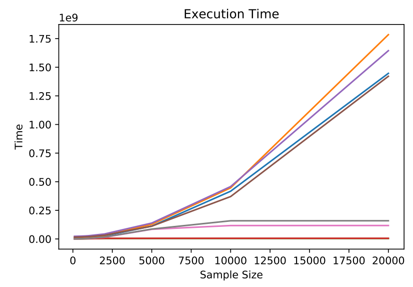
The RMS shown in Table III is calculated between the orange object shown in Fig 6b with the point cloud generated from the CAD that gave the smallest RMS. The same point cloud gave the lowest RMS in all cases except for SHOT, where one point cloud gave a lower RMS. Using RMS is not an accurate measurement for classification, but it was sufficient for this experiment. As shown in the table, FPFH is the fastest of the methods. However, it failed to give an accurate estimate, since the estimate was flipped upside down. A note on the PPF estimate is that the resulting pose estimate from the method had the two point clouds very far from each other, about 10 cm on average, and only by using the ICP method, did it achieve



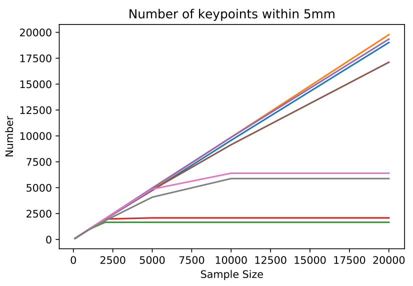
(a) Angle error [rad], measured using an angle-axis calculation of the error transformation



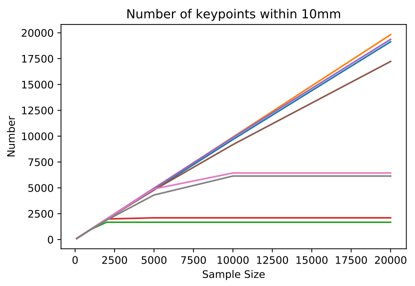
(b) Distance error [mm], measured using the distance of the error transformation



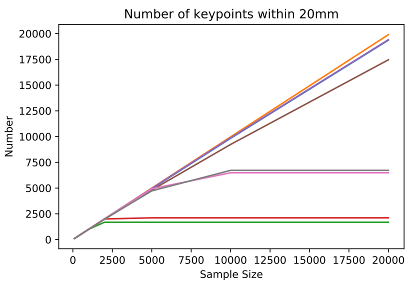
(c) Execution time [µs]



(d) Number of times where the estimated correspondence was within $5\,\mathrm{mm}$ away from the correct corresponding point



(e) Number of times where the estimated correspondence was within $10\,\mathrm{mm}$ away from the correct corresponding point



(f) Number of times where the estimated correspondence was within $20\,\mathrm{mm}$ away from the correct corresponding point

Fig. 10. Results when adjusting the number of keypoints used parameter. Each coloured graph represents the results from a sample point cloud, where 8 are selected from the total of 129 point clouds. The red, green and orange graphs are of different viewpoint point clouds of part A, the blue, purple and pink graphs are of different viewpoint point clouds of part B, and the brown and gray graphs are two point clouds of the Stanford Bunny.

a more accurate pose. The resulting pose estimation can be seen in Fig 11.

V. Discussion

Experiment 1 tries to evaluate the parameter range that achieves the best results, and the results are promising. The point clouds that were tested are similar to some degree, especially since the majority of point clouds are of two objects. Because of this it is not possible to conclude the optimal parameters before the method has been tested on a wider range of point clouds.

It is not fair to compare the execution time of the proposed method to the ones provided by the PCL library. The code used for the proposed method is not yet designed for optimization, and can in most cases be improved.

For instance, in the preprocessing step Section III-B1, each point in the point cloud checks which points are within a given radius. In the code, the method goes through every point for each point, making it an algorithm with a complexity of $O(n^2)$. This could be greatly improved by using smarter methods such as k-d tree structures or similar methods, which could lower the complexity to $O(kN^{1-\frac{1}{k}})$ where k is the dimension of the tree.

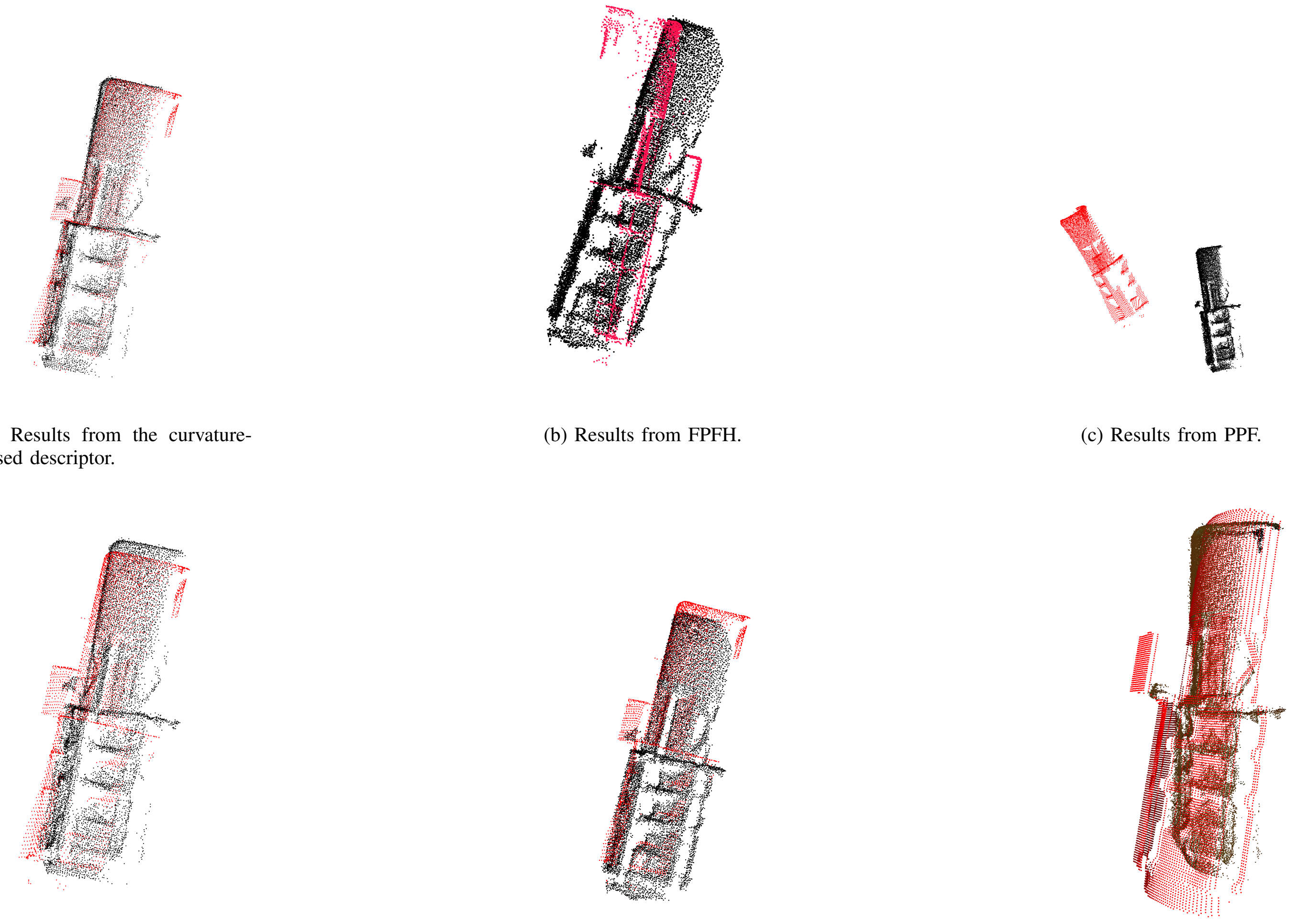
Though the proposed method presents many equations which uses Conformal Geometric Algebra, the implementation could benefit from using linear algebra and matrix manipulation, as this is more computationally efficient. The least square optimization in Section III-D for instance, is a general optimization algorithm that encompasses all Conformal Geometric Algebra objects. Since the proposed method only uses points, it would be beneficial to use a linear algebra least square optimization algorithm such as [2], [37]

It is also worth noting that the results from the GADS method, could probably be improved, if the point clouds would have color information. This was not possible in the experiments because of the CAD models, which did not have any color.

The proposed method does not handle scaling. This is because the spheres in the descriptor are specified with a radius. In order to make the method scale-invariant, the descriptors requires a reference scale, which could be developed.

VI. Conclusion

The proposed method uses an analytic approach to generating descriptors, using Conformal Geometric Algebra. The descriptor consists of two spheres which represents the curvature surrounding a point. This method was compared with a selection of some state-of-the-art methods, and the results were presented. In the experiment where the point cloud was compared to itself, the proposed method and PPF generated the most accurate results, while in the experiment where a CAD



(a) Results from the curvature-based descriptor.

(b) Results from FPFH.

(c) Results from PPF.

(d) Results from SHOT.

(e) Results from 3DSC.

(f) Results from GASD.

Fig. 11. A sample of the resulting position estimation with the different descriptor methods. The red points are from the CAD model, while the black points are taken with the Zivid camera.

model point cloud was compared to an object captured with a 3D camera, the proposed method showed that the accuracy and robustness was sufficient for it to be used in industrial applications.

References

[1] A. L. Alexander, K. Hasan, G. Kindlmann, D. L. Parker, and J. S. Tsuruda. A geometric analysis of diffusion tensor measurements of the human brain. *Magnetic Resonance in Medicine*, 44(2):283–291, 2000.

[2] K. Arun, T. Huang, and S. Blostein. Least-squares filtering of two 3-d point sets. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-9(5):698 – 700, 1987.

[3] P. Besl and N. McKay. A Method for Registration of 3-D Shapes, 1992.

[4] S. Bouaziz, A. Tagliasacchi, and M. Pauly. Sparse iterative closest point. *Computer Graphics Forum*, 32(5):113–123, 2013.

[5] U. Castellani and A. Bartoli. 3D shape registration. *3D Imaging, Analysis and Applications*, 9781447140:221–264, 2014.

[6] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Proceedings. 1991 IEEE International Conference on Robotics and Automation*, 10(3):2724–2729, 1991.

[7] C. S. Chua and R. Jarvis. Point Signatures: A New Representation for 3D Object Recognition. *International Journal of Computer Vision*, 25(1):63–85, 1997.

[8] D. H. Chung, I. D. Yun, and S. U. Lee. Registration of Multiple Range Views using the Reverse Calibration Technique. *Pattern Recognition*, 31(4):459–464, 1997.

[9] P. Colapinto. Versor: Spatial computing with conformal geometric algebra. Master's thesis, University of California at Santa Barbara, 2011. Available at http://versor.mat.ucsb.edu.

[10] L. Dorst. Total Least Squares Fitting of k-Spheres in n-D Euclidean Space Using an (n+2)-D Isometric Representation. *Journal of Mathematical Imaging and Vision*, 50(3):214–234, 2014.

[11] L. Dorst, D. Fontijne, and S. Mann. *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2009.

[12] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3D object recognition. In *Proceedings of*

the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 998–1005, 2010.

[13] A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing Objects in Range Data Using Regional Point Descriptors. In *European Conference on Computer Vision*, volume 3023, pages 224–237, 2004.

[14] D. Girardeau-Montaut, M. Roux, R. Marc, and G. Thibault. Change detection on points cloud data acquired with a ground laser scanner. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(3):W19, 2005.

[15] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[16] D. Hildenbrand. *Foundations of Geometric Algebra Computing*, volume 8 of *Geometry and Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[17] S. Hinterstoisser, V. Lepetit, N. Rajkumar, and K. Konolige. Going further with point pair features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9907 LNCS:834–848, 2016.

[18] H. Ho and D. Gibbins. Curvature-based approach for multi-scale feature extraction from 3D meshes and unstructured point clouds. *IET Computer Vision*, 3(4):201, 2009.

[19] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

[20] A. L. Kleppe and O. Egeland. A Curvature-Based Descriptor for Point Cloud Alignment using Conformal Geometric Algebra. *Advances in Applied Clifford Algebras*, 28(2):50, 2018.

[21] S. U. C. G. Laboratory. Stanford point cloud database. http://graphics.stanford.edu/data/3Dscanrep, 1994.

[22] H. Li and R. Hartley. The 3D-3D registration problem revisited. *Proceedings of the IEEE International Conference on Computer Vision*, 2007.

[23] J. P. S. d. M. Lima and V. Teichrieb. An Efficient Global Point Cloud Descriptor for Object Recognition and Pose Estimation. In *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 3, page 56–63, 2016.

[24] F. Mohideen and R. Rodrigo. Curvature Based Robust Descriptors. *Procedings of the British Machine Vision Conference 2012*, pages 1–41, 2012.

[25] V. Pratt. Direct least-squares fitting of algebraic surfaces. *ACM SIGGRAPH Computer Graphics*, 21(4):145–152, 1987.

[26] M. Ritter, W. Benger, B. Cosenza, K. Pullman, H. Moritsch, and W. Leimer. Visual Data Mining Using the Point Distribution Tensor. In *ICONS*, pages 10–13, 2012.

[27] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM*, 2001-Janua:145–152, 2001.

[28] R. B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. *IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009.

[29] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram. In *Intelligent Robots and Systems (IROS),*, pages 2155–2162, 2010.

[30] R. B. Rusu and S. Cousins. 3D is here: point cloud library. *IEEE International Conference on Robotics and Automation*, pages 1 – 4, 2011.

[31] B. Sabata and J. K. Aggarwal. Surface Correspondence and Motion Computation from a Pair of Range Images. *Computer Vision and Image Understanding*, 63(2):232–250, 1996.

[32] J. Salvi, C. Matabosch, D. Fofi, and J. Forest. A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing*, 25(5):578–596, 2007.

[33] S. Skotheim, H. Schumann-Olsen, and et. al. ZividLabs. Zivid 3d camera.

[34] A. Sveier, A. L. Kleppe, L. Tingelstad, and O. Egeland. Object Detection in Point Clouds Using Conformal Geometric Algebra. *Advances in Applied Clifford Algebras*, 27(3):1–16, 2017.

[35] L. Tingelstad and O. Egeland. Motor parameterization. *Advances in Applied Clifford Algebras*, 28(2):34, Mar 2018.

[36] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6313 LNCS(PART 3):356–369, 2010.

[37] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[38] R. Valkenburg and L. Dorst. Estimating Motors from a Variety of Geometric Data in 3D Conformal Geometric Algebra. *Guide to Geometric Algebra in Practice*, XVII(December):25–45, 2011.

[39] J. Yang, H. Li, and Y. Jia. Go-ICP: Solving 3D registration efficiently and globally optimally. *Proceedings of the IEEE International Conference on Computer Vision*, pages 1457–1464, 2013.

Adam Leon Kleppe Adam Leon Kleppe received his MSc in Cybernetics at the Norwegian University of Science and Technology (NTNU), Trondheim, Norway in 2013. He is currently working on his PhD also at the Norwegian University of Science and Technology. He focuses on 3D computer vision technology and how it can be improved to be used for assembly operations in the automotive industry.

Lars Tingelstad recieved the M.Sc. and Ph.D. degree in Mechanical Engineering from the Norwegian University of Science and Technology, NTNU, in 2011 and 2017, respectively. His Ph.D. thesis was on the estimation of rigid body motions from observation of 3-D geometric objects such as points, lines, planes, circles, and spheres, in conformal geometric algebra. He is currently employed as a researcher at the department of Mechanical and Industrial Engineering, NTNU, working on the research program SFI Offshore Mechatronics funded by the Norwegian Research Council.

Olav Egeland graduated with a MSc (1984) and a PhD (1987) in automatic control from the Norwegian University of Science and Technology (NTNU). He was professor of automatic control from at NTNU from 1989 to 2004, and was co-founder of a start-up from 2004 - 2011. He is currently professor of production automation at NTNU. He received the Automatica Prize Paper Award (1996) and the IEEE Trans. Control System Technology Outstanding Paper Award (2000). He was Associate Editor of IEEE Trans. Automatic Control (1996-1999) and European Journal of Control (1998-2000). His research interests are within mathematical modeling, robotic production, and offshore control systems.