Object Detection in Point Clouds Using Conformal Geometric Algebra

Aksel Sveier *, Adam Leon Kleppe, Lars Tingelstad and
Olav Egeland

Abstract. This paper presents an approach for detecting primitive geo-
metric objects in point clouds captured from 3D cameras. Primitive
objects are objects that are well defined with parameters and math-
ematical relations, such as lines, spheres and ellipsoids. RANSAC, a
robust parameter estimator that classifies and neglects outliers, is used
for object detection. The primitives considered are modeled, filtered
and fitted using the conformal model of geometric algebra. Methods
for detecting planes, spheres and cylinders are suggested. Least squares
fitting of spheres and planes to point data are done analytically with
conformal geometric algebra, while a cylinder is fitted by defining a non-
linear cost function which is optimized using a nonlinear least squares
solver. Furthermore, the suggested object detection scheme is combined
with an octree sampling strategy that results in fast detection of multi-
ple primitive objects in point clouds.

1. Introduction

There has been an increase in the use of 3D cameras in robotic vision appli-
cations due to the availability of commercial products with high accuracy.
The advantage of 3D cameras compared to 2D cameras is the additional
depth information, which can provide information about size and position of
objects in a scene, where a scene is the environment captured by the camera.
The depth information from 3D cameras can be represented as point clouds,
which is a set of points in Euclidean space given by the x, y, z coordinates
for each point. Object detection in point clouds can be difficult due to noise,
outliers and complexity in the data.

There exists several approaches for describing the underlying features
and geometry represented in a point cloud. In [10] features such as curvature,
surface normals and edges are extracted and objects are described in terms

---

*Corresponding author.

of such features. In [2] the focus is on extracting edges by constructing a covariance matrix for each point. In [13] a point distribution tensor is used, which is similar to calculating the covariance matrix, to describe regions of linear, planar and isotropic structure. These methods are well suited for describing arbitrary and complex geometry as sets of features.

Methods such as RANSAC [6] and the Hough transform [12] are well suited for detecting primitive objects in point clouds, such as lines, planes, spheres and cylinders. RANSAC is an iterative method for estimating parameters of a mathematical model from experimental data, while the Hough transform maps points to the parameter space, and classification is done based on voting schemes. Parameter space is in this context defined as the set of all possible combinations of parameters of a mathematical model. These methods handles outliers well, and has been widely used to recognize primitive objects in 2D and 3D images. Computer software libraries such as PCL [14] and OpenCV [3] contain a wide range of methods for detecting objects in point clouds.

In [15], the authors show that a general point cloud can be described accurately with a set of primitive shapes. They organize the point cloud in an octree structure, which is a recursive division of 3D space into octants. They continue by applying RANSAC within each subspace of the octree for five different primitives (plane, sphere, cylinder, cone, torus). This allows them to transform the data set from a point cloud to a set of primitive shapes. The number and variety of primitives depend on the geometry represented by the point cloud.

In this paper we focus on the detection of primitive geometric models in point clouds using RANSAC. In contrast to [15], we introduce the framework of the conformal model of geometric algebra [5,9], so that primitives can be constructed in a simple manner. Furthermore, a combination of these primitives in the conformal model can be used to construct a cylinder.

A central step in the RANSAC algorithm is to classify inliers and outliers. We show that conformal geometric algebra (CGA) enable filters with geometrical interpretation for inlier/outlier classification. The last step of the RANSAC algorithm is fitting the primitive to its inliers. This can be performed analytically with CGA, and the method is identical for both planes and spheres. Inspired by [15], we show that our suggested object detection method can be extended to multiple object detection, based on an octree sampling strategy.

The paper is organized as follows: Sect. 2 introduces the conformal model of geometric algebra, least square fitting in conformal space, RANSAC and octrees, Sect. 3 describes how the considered primitives are constructed from point data with CGA, followed by how inlier classification and scoring is implemented in CGA. Section 4 shows how octrees can be used to find multiple primitive objects in a point cloud. Section 5 present the experimental results, and finally Sect. 6 concludes the paper

2. Preliminaries

2.1. Conformal Geometric Algebra

The geometric algebra of the Euclidean space is denoted $\mathbb{R}_3$, while the conformal model of geometric algebra is denoted $\mathbb{R}_{4,1}$, which is the geometric algebra used in the conformal space $\mathbb{R}^{4,1}$. A basis of the conformal space is $\{e_0, e_1, e_2, e_3, e_\infty\}$. This basis is referred to as a null basis because $e_\infty^2 = e_0^2 = 0$. The basis vector e_∞ represents the point at infinity, while the basis vector e_0 represents an arbitrary origin. The inner product of these basis vectors is $e_\infty \cdot e_0 = -1$.

A blade in geometric algebra is the term used for any outer product of vectors. A vector is equivalent to a 1-blade, while a scalar is a 0-blade. A 2-blade is the outer product of two vectors $A \wedge B$, while the general k-blade is the outer product of k vectors. The grade of a k-blade is k. A multivector is the general term for a sum of blades.

The notation $\mathbb{R}_3^k$ refers to the k-grade elements of $\mathbb{R}_3$. The highest grade element of $\mathbb{R}_3$, the Euclidean pseudoscalar, is of grade 3 and is denoted $\boldsymbol{I}_3$. The conformal pseudoscalar is denoted $\boldsymbol{I}$ and is of grade 5. In this paper the representation of geometric objects and their duals is based on the formulation in [5], where a geometric object has the direct form $\boldsymbol{X}$ and the dual form $\boldsymbol{X}^* = \boldsymbol{X} \cdot \boldsymbol{I}^{-1}$. Note that an alternative notation is used in, e.g., [7], where the direct form is termed the IPNS representation, and the dual is the OPNS representation.

Vectors $\boldsymbol{p} \in \mathbb{R}_3$ maps to points $\boldsymbol{P} \in \mathbb{R}_{4,1}$ using

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

The inner product between two points is

$$\boldsymbol{P}_1 \cdot \boldsymbol{P}_2 = \frac{1}{2}(\boldsymbol{p}_2 - \boldsymbol{p}_1)^2 \tag{2}$$

which means that it is dependent on the distance between the Euclidean points.

Lines $\boldsymbol{L} \in \mathbb{R}_{4,1}^3$ are constructed through the outer product of two conformal points and the point at infinity

$$\boldsymbol{L} = \boldsymbol{P}_1 \wedge \boldsymbol{P}_2 \wedge e_\infty. \tag{3}$$

Planes $\boldsymbol{\Pi} \in \mathbb{R}_{4,1}^4$ are constructed through the outer product of three conformal points and the point at infinity

$$\boldsymbol{\Pi} = \boldsymbol{P}_1 \wedge \boldsymbol{P}_2 \wedge \boldsymbol{P}_3 \wedge e_\infty. \tag{4}$$

Unit dual planes are denoted $\boldsymbol{\Pi}^* \in \mathbb{R}_{4,1}^1$ and defined as

$$\boldsymbol{\Pi}^* = \hat{\boldsymbol{n}} + de_\infty \tag{5}$$

where $\hat{\boldsymbol{n}}$ is the normal of the plane and d is the distance from the plane to the origin along $\hat{\boldsymbol{n}}$.

Spheres $\boldsymbol{S} \in \mathbb{R}_{4,1}^4$ are constructed through the outer product of four points where at least one is not coplanar.

$$\boldsymbol{S} = \boldsymbol{P}_1 \wedge \boldsymbol{P}_2 \wedge \boldsymbol{P}_3 \wedge \boldsymbol{P}_4 \tag{6}$$

Dual spheres are denoted $\boldsymbol{S}^* \in \mathbb{R}^1_{4,1}$ and defined as

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

where $\boldsymbol{P}$ is the center point of the sphere, and ρ is the sphere radius. The radius of a sphere can be found with the inner product, where $\boldsymbol{S}^* \cdot \boldsymbol{S}^* = \rho^2$.

Circles are given by

$$\boldsymbol{C} = \boldsymbol{P}_1 \wedge \boldsymbol{P}_2 \wedge \boldsymbol{P}_3. \tag{8}$$

In order to find the radius of the circle, a sphere which has the circle as an equator can be constructed

$$\boldsymbol{S}^* = \frac{\boldsymbol{C}}{\boldsymbol{C} \wedge e_\infty} \tag{9}$$

The radius can then be found using $\boldsymbol{S}^* \cdot \boldsymbol{S}^* = \rho^2$.

The dual of a circle $\boldsymbol{C}^*$ is the outer product of two dual spheres $\boldsymbol{C}^* = \boldsymbol{S}^*_1 \wedge \boldsymbol{S}^*_2$.

2.2. Least Squares Fitting in Conformal Space

Least squares fitting procedure of points to spheres and planes in conformal space [8] can be obtained by minimizing the error function

$$\min \sum_{i=1}^{n}(\boldsymbol{P}_i \cdot \boldsymbol{X}^*)^2 \tag{10}$$

where $\boldsymbol{X}$ is either a plane or a sphere in the conformal model. The inner product $\boldsymbol{P}_i \cdot \boldsymbol{X}^*$ is a distance measure between a point $\boldsymbol{P}_i$ and the plane or sphere $\boldsymbol{X}^*$.

The equations can be written in a bilinear form:

$$\min(x^T \mathbf{B} x), \qquad x = (x_1, x_2, x_3, x_4, x_5)^T, \tag{11}$$

where $\mathbf{B}$ is a symmetric 5×5 matrix and the entries are

$$b_{j,k} = \sum_{i=1}^{n} w_{i,j} w_{i,k}, \qquad w_{i,k} = \begin{cases} p_{i,k} & \text{if } k \in \{1,2,3\} \\ -1 & \text{if } k = 4 \\ -\frac{1}{2}\mathbf{p}_i^2 & \text{if } k = 5. \end{cases}$$

It can be shown that the solution of the minimization problem is given by the eigenvector of $\mathbf{B}$ which corresponds to the smallest eigenvalue [8]. Since $\mathbf{B}$ is symmetric, the eigenvector can be found by singular value decomposition

$$\mathbf{B} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T \tag{12}$$

where $\mathbf{U}$ is a matrix where each column represents an eigenvector of $\mathbf{B}$, and $\boldsymbol{\Sigma}$ is a diagonal matrix where each diagonal value represent the corresponding eigenvalue of the eigenvectors in $\mathbf{U}$.

2.3. RANSAC

RANdom SAmple Consenus (RANSAC) [6] is an iterative method for estimating model parameters from a data set containing several outliers. For each iteration the steps of the RANSAC algorithm are as follows

1. Randomly sample a minimal subset of points required to generate a model. This is called the model candidate.
2. Categorize all the data points either as inliers or outliers. Inliers support the model candidate, while outliers oppose the model candidate.
3. If the model candidate has more inliers than the current best model estimate, it becomes the current best model estimate.

In addition to the data set and the specified model, the RANSAC algorithm requires three parameters:

- The error tolerance E_T, used to determine whether or not a point is an inlier.
- The number of iterations, k.
- The threshold t, which is the number of compatible points used to imply that the correct model has been found.

The iteration is terminated if a candidate has more than t inliers or after k iterations. After the last iteration, the model estimate is fitted to its inliers using least squares techniques.

2.3.1. Performance.   Performance considerations [15] of the RANSAC algorithm can be made by assuming a point cloud $\mathcal{P}$ of N points containing a object X of n points. A subset of q points is randomly sampled, where q is the minimum number of points that will define the object. The probability of detecting the object X in a single pass of the RANSAC algorithm is:

$$P(n) = \frac{\binom{n}{q}}{\binom{N}{q}} \approx \left(\frac{n}{N}\right)^q.$$
(13)

The probability of detecting the object after k iterations is:

$$P(n,k) = 1 - (1 - P(n))^k.$$
(14)

The number of iterations K required to detect a object of size n with a probability $P(n,K) \geq p_t$:

$$K \geq \frac{\ln(1 - p_t)}{\ln(1 - P(n))},$$
(15)

where p_t is the probability of detection given K iterations. The denominator can be approximated by its Taylor series if $P(n)$ is sufficiently small:

$$K \approx \frac{-\ln(1 - p_t)}{P(n)}.$$
(16)

Thus, the number of iterations required to detect a object is directly correlated to the fraction $\frac{n}{N}$.

2.4. Octree

Octrees are used to recursively subdivide 3D space into octants. Each octant is called a node, and each node can be subdivided into exactly eight nodes at the lower level of the octree. The largest node is called the root node and contain all other nodes.

By placing the root node of an octree over the space spanned by a point cloud, the points in the point cloud is organized into the nodes of the octree. The multiple levels of the octree allows for systematic searching of different sized point clusters in the point cloud.

The depth h of the octree is the number of levels of the octree. The resolution h_{res} of an octree is the length of the sides of the nodes at the lowest level of the octree. An octree is fully defined when the length l of the sides of the root node is specified along with the depth or resolution. The relation between the root node, depth and resolution can be expressed as:

$$\frac{l}{2^h} = h_{\mathrm{res}}. \tag{17}$$

The number of nodes n_n present in an octree is:

$$n_n = 1 + 8^1 + 8^2 + \cdots + 8^{h-1}. \tag{18}$$

3. Object Detection

RANSAC can be implemented for object detection of primitives in point clouds using the Conformal model of Geometric Algebra as the mathematical framework. Planes, spheres and cylinders are considered and the steps are described below.

3.1. Planes and Spheres

A minimal sub-sample of q points are randomly sampled from the point cloud. Planes and spheres are constructed according to (4) and (6), where $q = 3$ and $q = 4$, respectively. This means that no further processing is required to construct the plane or sphere.

3.2. Cylinders

A cylinder cannot be described by a blade in the conformal model, instead a circle–line representation can be used. The line describes the center-axis of the cylinder, while the circle radius describes the radius of the cylinder given that the circle is parallel and coinciding with the center-axis. Two approaches for constructing a cylinder from point data are considered and illustrated in Fig. 1

1. A sub-sample of 3 random points, $\boldsymbol{P}_1$, $\boldsymbol{P}_2$ and $\boldsymbol{P}_3$ is sampled and a circle is constructed from the points through the relation $\boldsymbol{C} = \boldsymbol{P}_1 \wedge \boldsymbol{P}_2 \wedge \boldsymbol{P}_3$. The cylinder's radius is the radius of the circle and can be found by constructing a sphere that has the circle as its equator, according to (9). The cylinder axis, which goes through the circle center along the normal of the plane that the circle lies on, is defined as $L = \boldsymbol{C}^* \wedge e_\infty$.
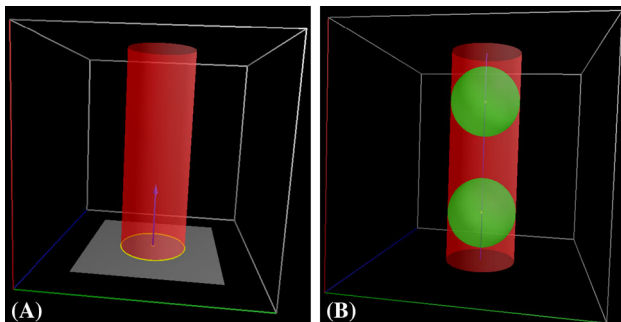
FIGURE 1. The two approaches considered for constructing a cylinder. **a** *Circle–line* construction, **b** *sphere–sphere* construction

2. Two spheres are subsequently fitted to the cylinder using RANSAC for spheres. The center of the spheres $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ will coincide with the cylinder center-axis. Each sphere is constructed from four points sampled from the cylinder surface. This approach is potentially more robust than the one mentioned above, as it uses more information to detect the cylinder. The cylinder center-axis is described by the line $L = \boldsymbol{S}_1^* \wedge \boldsymbol{S}_2^* \wedge e_\infty$, which describes the line connecting the centers of the two spheres. The cylinder radius is set to be the average of the radii of the two spheres $\rho = \frac{\rho_1 + \rho_2}{2}$.

3.3. Inlier Classification

Inlier classification in RANSAC is based on the distance from a point to the model surface. In conformal space, the inlier points that belong to a primitive can be classified using a filter. The filter classifies all points $\boldsymbol{P}_i \in \mathcal{P}$ that lie within the distances $d_{\min}$ and $d_{\max}$ from an object as inliers to that object.

For a plane, the filter is formulated as

$$d_{\min} \leq \boldsymbol{P}_i \cdot \boldsymbol{\Pi}^* \leq d_{\max} \tag{19}$$

where $d_{\min}$ and $d_{\max}$ are the distance from the plane $\boldsymbol{\Pi}$. The points that satisfies the equation are kept, while the others are discarded.

A similar approach can be used for spheres

$$d_{\min} \leq 2\sqrt{-\boldsymbol{P}_i \cdot \boldsymbol{P}} \leq d_{\max} \tag{20}$$

where $\boldsymbol{P}$ describes the sphere center and $d_{\min}$ and $d_{\max}$ are the minimum and maximum radius of the sphere, respectively.

For cylinders, the expression is

$$d_{\min} \leq 2\sqrt{-\boldsymbol{P}_i \cdot \frac{\boldsymbol{P}_i \wedge \boldsymbol{L}^*}{\boldsymbol{L}^*}} \leq d_{\max} \tag{21}$$

where $\boldsymbol{L}$ describes the cylinder center axis and $d_{\min}$ and $d_{\max}$ are the minimum and maximum radius of the cylinder, respectively. The expression

$$\frac{\boldsymbol{P}_i \wedge \boldsymbol{L}^*}{\boldsymbol{L}^*} \tag{22}$$

describes a sphere with center coinciding with the cylinder center axis $\boldsymbol{L}$ and the point $\boldsymbol{P}_i$ lying on its equator.

3.4. Scoring

Once the inliers for a candidate are identified, the candidate is assigned a fitness score $f(I)$, where I is the number of inliers. The fitness function is used to determine if the proposed model candidate is a valid model, or if it is a subject to noise/outliers. The candidate is valid if $f(I) \geq t$, where t is the threshold parameter presented in Sect. 2.3.

The fitness function for planes is defined as

$$f_\Pi(I) = I. \tag{23}$$

This means that iteration will continue until a plane with score $f_\Pi \geq t$ is found. If no such plane is found the algorithm will terminate after k iterations and the highest scoring plane will be returned as the plane estimate.

Since spheres have finite surface areas, the number of points that lies on a sphere surface is related to the radius ρ and its distance z from the sensor. This is used to calculate an estimate of inliers for a certain sphere, which is compared to the actual number of inliers to decide the sphere's fitness. The estimate $\hat{I}$ is defined as

$$\hat{I} = n_A A_S \tag{24}$$

where n_A is the number of points in the point cloud per unit area in relation to the z-distance from the sensor and A_S is the surface area of the sphere. This gives

$$\hat{I} = a r^2 z^b \tag{25}$$

Here, r is the radius of the sphere, z is the z-distance of the sphere center from the sensor and a and b are constant parameters that describes the density of points in relation to the distance from the sensor. The a and b parameters are sensor specific and could be found experimentally by placing three spheres with different radius in the field of view of the camera. The inliers for each sphere are then measured, along with their distance from the sensor and their radius. The measurements could then be fitted to $\hat{I}$ by optimizing a and b. For the camera used in the experiments, these parameters were found to be $a = 439731.612$ and $b = -2.234$.

The score is calculated as the relation between the estimated inliers and the actual inliers

$$f_S(I) = \frac{I}{\hat{I}} \tag{26}$$

The algorithm will terminate and the sphere estimate will be returned when a score above the threshold t is achieved or the number of iterations k is reached.

Cylinders can have a finite surface area if the cylinder length is determined. Cylinders are, however, not invariant to rotation as spheres are. This

means that in order to find the inlier estimate, one must first determine the orientation of the cylinder, followed by the area which is visible by the camera. Thus, for simplicity, a cylinder is scored solely based on the number of inliers I that fall on the cylinder surface

$$f_G(I) = I \tag{27}$$

Consequently, the cylinder model candidate which has the highest number of inliers will be accepted as the cylinder estimate, which is usually the largest cylinder in the point cloud.

3.5. Fitting

Planes and spheres are fitted using the analytic least squares method described in Sect. 2.2. Cylinders are fitted with nonlinear least squares methods using the optimization tool [17], which is a software library that enables optimization through automatic differentiation of conformal entities [18] and the Ceres Solver [1] by Google.

The cost-function to be minimized is defined as

$$\min \sum_{i=1}^{n} \left(\sqrt{ -\frac{\boldsymbol{P}_i \wedge \boldsymbol{L}^*}{\boldsymbol{L}^*} \cdot \boldsymbol{P}_i } - \rho \right)^2 \tag{28}$$

which describes the sum of squared distance between a set of points $\boldsymbol{P}_i \in \mathcal{P}$ and a cylinder with center axis $\boldsymbol{L}$ and radius ρ. The expression $\frac{\boldsymbol{P}_i \wedge \boldsymbol{L}^*}{\boldsymbol{L}^*}$ describes a sphere with the center coinciding with the line $\boldsymbol{L}$ and the point $\boldsymbol{P}_i$ lying on its equator. Taking the inner product between this sphere and the point $\boldsymbol{P}_i$ gives the negative squared distance between the point and the line. The minimal sum of squared distances is found by optimizing $\boldsymbol{L}$ and ρ.

4. Multiple Object Detection

Using the presented RANSAC based object detection method as a foundation, multiple objects can be detected in a point cloud by organizing it in an octree. A method is suggested based on the sampling strategy of Schnabel et al. [15].

4.1. Overview

Given a point cloud $\mathcal{P}$ of points $\{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_N\}$, the point cloud is organized in an octree structure. The size of the octree is set such that the root node of the octree spans over the whole point cloud. The root node is subdivided into smaller nodes until nodes with sides of length h_{res} are obtained. The resolution h_{res} of the octree should be chosen such that there exist a node that contains solely inlier points of the smallest object present in the point cloud.

Model candidates are constructed by picking sub-samples from within a single node. Inlier classification, scoring and fitting is performed as presented in Sect. 3. The subset of inliers for a object $\mathcal{P}_{\boldsymbol{X}_i} \in \mathcal{P}$ is extracted from the point cloud if the score is larger than some threshold t. Because there exist a node of solely inlier points for every object present in the point cloud, it is only necessary to iterate $k = 1$ time per node.
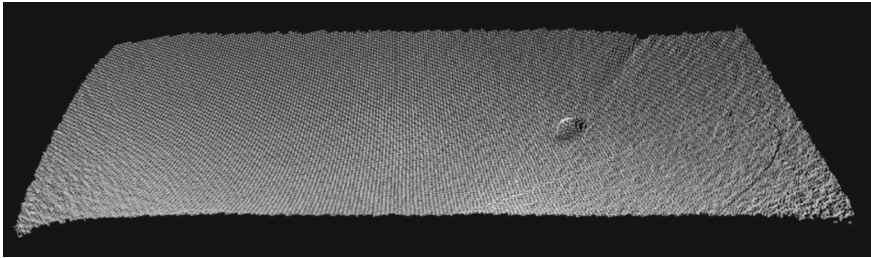
FIGURE 2. A point cloud of 49,396 points showing a ping-pong ball placed on the floor

By systematically repeating this procedure for every node at every level of the octree, all considered objects are detected. Furthermore, by starting the procedure at the highest level of the octree, large shapes are detected and extracted from the point cloud first. When the procedure is conducted for all the nodes in the octree, the extracted set of primitives $\boldsymbol{X} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_m\}$ are returned with their corresponding subsets of inliers $\mathcal{P}_{\boldsymbol{X}_1}, \ldots, \mathcal{P}_{\boldsymbol{X}_m} \in \mathcal{P}$.

4.2. Performance Considerations

The performance of this sampling strategy is best illustrated with an example. A point cloud of 49,396 points is shown in Fig. 2. The point cloud contains a sphere with radius $\rho = 0.02$ m and 109 inlier points, which is to be detected. The point cloud fits inside an octree with a root node with sides of 2.56 m. According to (18) a resolution of $h_{\mathrm{res}} = 0.01$ m suggest a total of 2,396,745 nodes, and setting $k = 2$ results in 4,793,490 iterations. In this case, only 10,118 nodes are occupied with more than four points, which is required to construct a sphere, resulting in a total of 20,236 iterations. Assuming that at least one of the nodes has a inlier ratio ≥ 0.95, detection of the sphere is achieved with a probability of 0.966. Detecting the sphere with the same probability using the random sampling scheme of the original RANSAC algorithm, would require 142,612,308,700 iterations according to (16). In this particular case the octree sampling scheme is more than 7,000,000 times more effective. This measure is highly dependent on the point density and the required depth of the octree.

5. Results

The suggested RANSAC based object detection methods were implemented in a C++ software application developed in the Qt Integrated Development Environment. The *Versor* [4] library was used for geometric algebra computations and PCL [14] was used for visualisations and handling point clouds. Experiments were conducted on a computer running the Ubuntu 14.04 LTS operating system with a Intel i7 3.50 GHz processor and 16 GB of memory.

The performance of the object detection methods was tested through an experiment where CAD-models of a sphere and a cylinder were sampled to point clouds and detection was performed. Noise/outliers were added to the

TABLE 1. Results for object detection of a sphere

Outliers (%)	Iterations	Time (ms)	Error (mm)	Probability of detection (%)
33	2	2.7	0.25	42.4
44	6	5.7	0.25	46.1
55	18	19.3	0.25	54.2
65	77	107.1	0.26	67.2
74	137	261.6	0.25	48.7
82	864	2588.4	0.26	62.2
90	6762	38634.9	0.81	51.8

The probability of detection is calculated according to (14)

sampled point clouds and detection was tested for several quantities of outliers. The number of iterations for successful detection was measured as the main performance metric, where a successful detection was defined as a object consisting of $\geq 90\%$ of the points of the original sampled sphere/cylinder. Moreover, the run-time of the object detection and the resulting geometrical error were measured. For a sphere this geometrical error was defined as the distance from the detected center to the true center, while for a cylinder the geometrical error were defined as the deviance in angle between the detected and true center axis. The variations in the results occurs due to the random nature of RANSAC, but consistent trends can be observed.

Detection of a sphere was tested first and the results are shown in Table 1. In addition to the metrics mentioned above, the probability of successful detection based on the number of iterations according to (14) is tabulated. The calculated probabilities lie in the range between 34.1 and 68.2%, which is the range of 1 standard deviation in a standard normal distribution. This implies that the probabilistic relations presented in Sect. 2.3.1 are valid for the implemented algorithm. It can be seen that there is a direct correlation between the number of iterations and the time of computation. In addition, the error metric was satisfactory for outlier levels up to 90%, implying that successful detections can be made in these conditions.

The detection of a cylinder was then investigated in an experiment where the two approaches of modeling a cylinder were considered, the results are shown in Tables 2 and 3 . In addition, the results are plotted in Fig. 3 with the outlier levels graphically illustrated. The sphere–sphere approach performed significantly better than the circle–line approach on all metrics. This implies that fitting spheres inside the cylinder gives a more accurate description of the cylinder axis than constructing it from three points on the cylinder surface. In order to accurately describe the cylinder axis from three points, the points have to be sampled from the cylinder surface and the plane they are describing needs to be normal to the center axis. The probability of achieving this with a random sampling strategy is intuitively small, especially with the presence of noise/outliers. Constructing two spheres from the surface of a cylinder

TABLE 2. Results for object detection of a cylinder

Outliers (%)	Iterations	Time (ms)	Error (°)
0	1988	273.8	1.257
34	13505	1683.9	1.452
51	14003	1961.6	1.906
64	21598	3706.7	1.369
75	54311	11592.0	1.996
91	8404	4539.8	10.498

The cylinder is initialized using the circle–line approach described in Sect. 3.1

TABLE 3. Results for object detection of a cylinder

Outliers (%)	Iterations	Time (ms)	Error (°)
0	557	11.9	0.657
31	1536	25.4	0.618
53	4035	69.5	0.913
65	6682	125.0	1.483
71	8567	188.5	1.298
91	63190	8652.8	8.949

The cylinder is initialized using the sphere–sphere approach described in Sect. 3.1

includes more information about the cylinder, as a total of eight points are used to describe the cylinder. Our results shows that this holds in practice. Good detections were achieved with up to 80% outliers.

The multiple object detection method was implemented for spheres and tested on a point cloud of 22,225 points showing three balls on a table. The point cloud was obtained with a Kinect for Xbox One 3D camera that output noisy point clouds. The Kinect have a time-of-flight depth sensor with a 512×424 pixel array with a pixel size of 10×10 μm. The accuracy error is 1% of the range for an average of 100 images [11]. The purpose was to test the algorithm when inliers are noisy and do not lie on a perfect sphere surface. Detection was performed 1000 times with an average run-time of 121 ms and 14,700 iterations. The results are presented in Table 4.

In the experiment all three spheres were detected 56.2% of the attempts. For 42.8% of the attempts, one or more spheres were not detected. Lastly, for 8.8% of the attempts false detections were made.

From Table 4 it can be seen that there is a correlation between detection rate and the number of inliers. An explanation can be that the scoring function is not valid for low amounts of inliers or that noise in the inliers causes poor representation of spheres with small radii/few inliers. For the largest sphere, the detected radius is significantly smaller than the actual
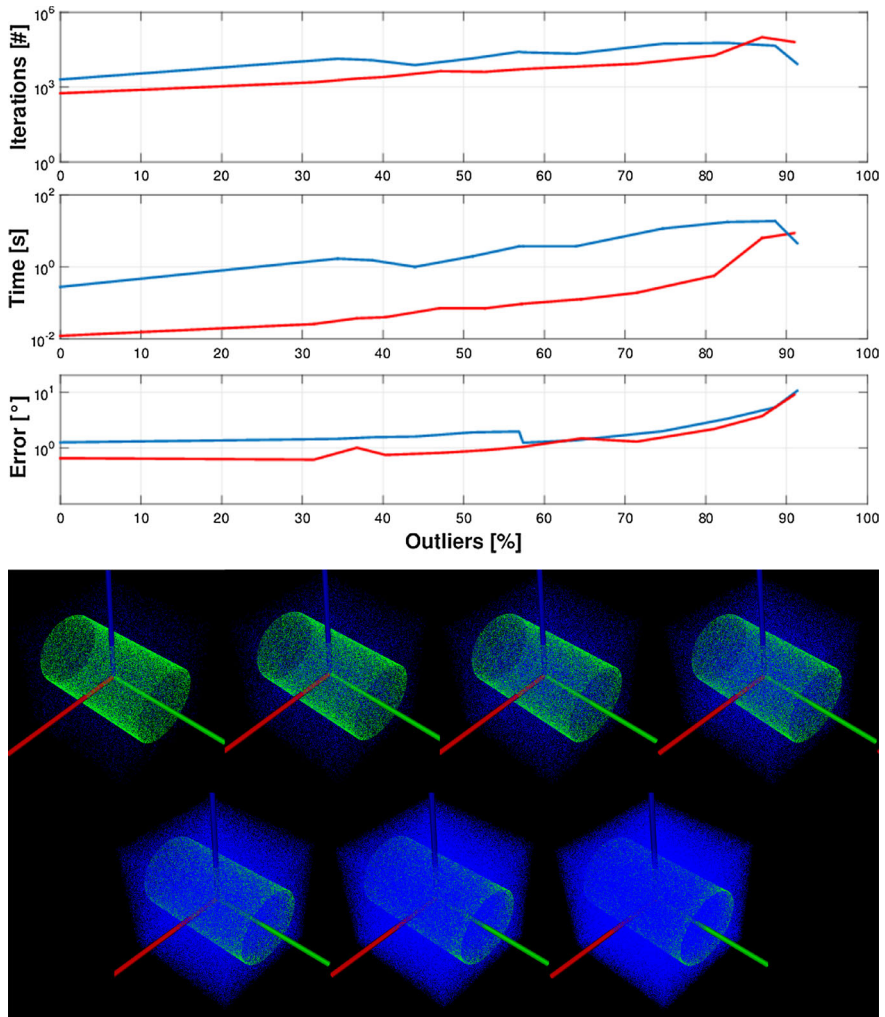
FIGURE 3. Detection of a cylinder with an increasing out-
liers percentage. The *circle–line* approach is plotted as the
blue line while the *sphere–sphere* approach is plotted as the
red. The performance of the algorithms is plotted logarith-
mically for an increasing outlier percentage. The outlier per-
centage is graphically illustrated below the plots

radius. This occurs because smaller spheres that are initialized on the sur-
face of the largest sphere are accepted. These spheres could be classified as
false detections, but are accepted because they are located correctly.

Detection rate can be improved by increasing the number of iterations
per node, at the cost of the run-time of the algorithm. False detections can
be avoided with a more robust scoring scheme.

TABLE 4. Performance metrics for the algorithm for detection of three different spheres in a point cloud

	Sphere 1	Sphere 2	Sphere 3
Inliers	202	141	73
Detection rate	0.989	0.872	0.707
Detected z-distance (m)	1.626	1.272	1.697
Uncertainty (mm)	6.81	4.79	5.74
Actual radius (mm)	33	20	20
Detected radius (mm)	28.94	18.97	19.85
Uncertainty (mm)	5.1	2.41	2.34
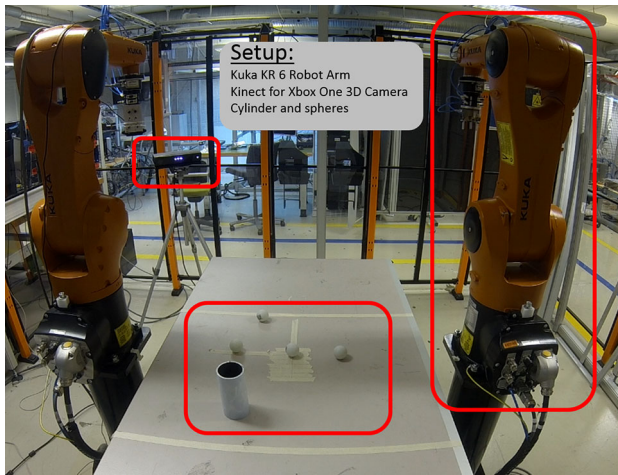
FIGURE 4. Setup of the robotic pick-and-place demonstration. Point clouds from the 3D camera is used for detecting the plane, spheres and cylinder. The information is sent to the robot arm, which is used to place the spheres in the cylinder

5.1. Demonstration

A robotic pick-and-place experiment was performed for demonstrating the suggested methods. A Kinect for Xbox One was used for obtaining point clouds of the scene and the primitive objects present were detected. Their positions and parameters were used for robot-camera calibration and robot positioning. The goal was to pick-and-place arbitrarily positioned ping-pong balls in a cylindrical tube. The setup of the demonstration can be seen in Fig. 4.

A video of the demonstration can be found in [16]. From the video it becomes clear that the suggested methods are satisfactory in the sense of accuracy, speed and repetitiveness required for high accuracy robotic pick-and-place applications, given raw data from consumer grade 3D cameras.

6. Conclusion

We have outlined the procedure of the RANSAC algorithm and showed how this paradigm combine with conformal geometric algebra to develop a robust and geometrical intuitive object detection method for point clouds. Spheres were successfully detected in point clouds with up to 90% outliers and cylinders could successfully be detected in point clouds with up to 80% outliers. We suggested two methods for constructing a cylinder from point data using CGA and found that fitting two spheres to a cylinder gave performance advantages compared to constructing a circle and line from 3 points on the cylinder surface. For a outlier level of 53%, the sphere–sphere approach used 73% fewer iterations, 96% less computation time and produced 52% less geometrical error than the circle–line approach. In addition, an algorithm for detecting multiple objects have been suggested and tested, with good results.

References

[1] Agarwal, S., Mierle, K., et al.: Ceres Solver. http://ceres-solver.org/. Accessed Sept 2016

[2] Bazazian, D., Casas, J.R., Ruiz-Hidalgo, J.: Fast and robust edge extraction in unorganized point clouds. In: 2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA), pp. 1–8. IEEE, New York (2015)

[3] Bradski, G.: OpenCV (2000)

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

[5] Dorst, L., Fontijne, D., Mann, S.: Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry. Morgan Kaufmann Publishers Inc., San Francisco (2009)

[6] Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6), 381–395 (1981)

[7] Hildenbrand, D.: Foundations of Geometric Algebra Computing. Springer, New York (2013)

[8] Hildenbrand, D.: Foundations of geometric algebra computing. In: Fitting of Planes or Spheres to Sets of Points, Least-Squares Approach, chap. 5.3 (2013)

[9] Hildenbrand, D., Hitzer, E.: Analysis of point clouds: using conformal geometric algebra. In: Proceedings of Third International Conference on Computer Graphics Theory and Applications, vol. 5, pp. 1–6 (2008)

[10] Merigot, Q., Ovsjanikov, M., Guibas, L.J.: Voronoi-based curvature and feature estimation from point clouds. IEEE Trans. Vis. Comput. Graph. **17**(6), 743–756 (2011)

[11] Payne, A., Daniel, A., Mehta, A., Thompson, B., Bamji, C.S., Snow, D., Oshima, H., Prather, L., Fenton, M., Kordus,L., et al.: 7.6 a 512× 424 CMOS 3D time-of-flight image sensor with multi-frequency photo-demodulation up to 130 MHz and 2 GS/s ADC. In: 2014 IEEE International on Solid-State Circuits Conference Digest of Technical Papers (ISSCC), pp. 134–135. IEEE, New York (2014)

[12] Richard, P.E.H., Duda, O.: Use of the Hough Transformation to Detect Lines and Curves in Pictures. Commun. ACM **15**(1), 11 – 15 (1972)

[13] Ritter, M., Benger, W., Cosenza, B., Pullman, K., Moritsch, H., Leimer, W.: Visual data mining using the point distribution tensor. In: IARIS Workshop on Computer Vision and Computer Graphics-VisGra (2012)

[14] Rusu, R.B., Cousins, S.: 3D is here: point cloud library (PCL). In: 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–4. IEEE, New York (2011)

[15] Schnabel, R., Wahl, R., Klein, R.: Efficient RANSAC for point-cloud shape detection. Comput. Graph. Forum **26**(2), 214–226 (2007)

[16] Sveier, A.: Video Demonstration and Source Code for Experiments Conducted in Object Detection in Point Clouds Using CGA (2016). doi:10.5281/zenodo. 158974

[17] Tingelstad, L.: GAME—Geometric Algebra Multivector Estimation (2016). http://github.com/tingelst/game/

[18] Tingelstad, L., Egeland, O.: Automatic multivector differentiation and optimization. In: Advances in Applied Clifford Algebras, pp. 1–15 (2016)

Aksel Sveier, Adam Leon Kleppe, Lars Tingelstad and Olav Egeland
Department of Mechanical and Industrial Engineering, Faculty of Engineering
NTNU, Norwegian University of Science and Technology
Trondheim
Norway
e-mail: aksel.sveier@ntnu.no

Adam Leon Kleppe
e-mail: adam.l.kleppe@ntnu.no

Lars Tingelstad
e-mail: lars.tingelstad@ntnu.no

Olav Egeland
e-mail: olav.egeland@ntnu.no