Doctoral theses at NTNU, 2018:197

Adam Leon Kleppe

Point Cloud Registration for Assembly using Conformal Geometric Algebra

NTNU
Norwegian University of
Science and Technology

Doctoral thesis

Adam Leon Kleppe

ISBN 978-82-326-3184-1 (printed ver.)
ISBN 978-82-326-3185-8 (electronic ver.)
ISSN 1503-8181

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of
Philosophiae Doctor
Faculty of Engineering
Department of Mechanical and Industrial
Engineering

Doctoral theses at NTNU, 2018:197

NTNU
Norwegian University of
Science and Technology

Adam Leon Kleppe

Point Cloud Registration for Assembly using Conformal Geometric Algebra

Thesis for the Degree of Philosophiae Doctor

Trondheim, June 2018

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Engineering
Department of Mechanical and Industrial Engineering

ISBN 978-82-326-3184-1 (printed ver.)
ISBN 978-82-326-3185-8 (electronic ver.)
ISSN 1503-8181

Doctoral theses at NTNU, 2018:197

Printed by NTNU Grafisk senter

To Linn

Abstract

This thesis is a collection of five journal papers and one conference paper. The thesis is on pose alignment and point correspondence estimation for 3D point clouds, and inverse kinematics of industrial robots. The approaches proposed in this thesis are based on conformal geometric algebra, which is an extension of Euclidean geometry which enables efficient descrition of geometric objects such as line, plane and sphere geometry, as well the calculation of the intersection between such objects.

The thesis presents a novel approach for the initial alignment between two point clouds called the Curvature-Based Descriptor. The curvature-based descriptor is a descriptor which describes the local curvature around a point in the point cloud. The local curvature is expressed with two spheres generated using conformal geometric algebra. The thesis also presents preprocessing steps which are used to segment the point cloud to extract only the parts of the point cloud that are necessary for the alignment, and a keypoint extraction method which extracts certain points from the point cloud, making the point correspondence more accurate.

The inverse kinematics presented in this thesis is an analytic solution which uses conformal geometric algebra. The solution is presented for the Kuka KR6 R900 sixx robot and the Universal Robots UR5 robot. All singularities and all configurations are accounted for in the solutions.

The thesis has several experimental results. These experiments are presented in each paper, and show the results from various methods performing point cloud alignment. The results show that it is possible to achieve a sub-millimeter accuracy for position estimation of an object using state-of-the-art methods when using both 3D and 2D cameras combined. The results also show that the curvature-based alignment method, after applying the preprocessing steps presented in the thesis, achieve a sub-millimeter accuracy on its own, an accuracy that is not achieved with any of the other 3D alignment methods.

Contents

List of figures

Acknowledgements

I would like to thank my supervisor Olav Egeland for his dedication and resourcefulness. I would not be able to write this thesis without his guidance.

I would also like to thank my co-supervisor Amund Skavhaug for giving me expert advice about the thesis and life in general, and for actively following me on my journey.

Lars Tingelstad has been one of the pillars which I stand upon. His vast knowledge of geometric algebra and his implementations of it, has helped me to develop my methods as well as on the experimental results.

I would also like to thank my colleagues for making my work days a pleasant time. Thank you, Aksel Sveier and Eirik Njåstad, who have contributed to my work and have discussed ideas with me. I would also like to thank Asgeir Bjørkedal and Krirstoffer Larsen for their contribution to my work.

My family and friends have all supported me throughout my journey, and I thank you from the bottom of my heart. Especially Linn, whom I love above all.

I want to thank Balder, Birk and Frøya for always being happy to see me, and for always bringing a smile to my face. I thank you for all the licks and cuddles, that have encouraged me to keep going.

Finally, I would like to thank the coffee machines, who have provided me with a nearly endless supply of hot chocolate.

Chapter 1

Introduction

What did the beach say when the tide came in?
Long time no sea.

1.1 Background and Motivation

This thesis is part of the project "Energy saving transmission control for heavy duty vehicles" in collaboration with Kongsberg Automotive. The overall goal is to create more efficient heavy duty vehicles, and one of these aspects is to create more efficient servo clutches. This thesis specifically looks at how the production of these servo clutches can be improved using 3D cameras to detect the car parts so that they can be assembled with robots.

Currently, the assembly process at Kongsberg Automotive is done manually, and in order to increase their production, the whole or parts of the assembly process should be automated. A fully automated production line is more efficient and less prone to errors. In order to successfully implement an automated production line, the position of all the assembly parts has to be known at all times. If there is an error between the actual position of the part and the measured or estimated position, the assembly process might be stopped, or in worst case, the parts or even the robots may break.

A widely used technique to ensure accurate positioning of an object is to use fixtures. Fixtures are either clamps or trays that hold the objects in place, and are constructed so that the object is forced into a specific, known position. This works for objects that are standardized and produced in large volumes, but the car parts from Kongsberg Automotive change frequently depending on the specifications from the customer. The dimensions and mounting holes of the servo clutch are specific to different cars, which changes the behaviour of the assembly process almost weekly. This means that these clamps and trays would have to be changed as frequently, which is not a desirable outcome.

This thesis has these considerations in mind. The main task is to investigate methods that estimates the position of objects with the accuracy required to complete the assembly task, and that are sufficiently flexible so that they can be

changed depending on the specifications from the customer. The company has provided a set of car parts and their CAD models, and the task is to estimate the position of these objects with a sub-millimeter accuracy based only on the provided CAD model.

1.1.1 Computer Vision

Computer vision is a broad term that encompasses all methods and technologies that enables computers to analyze sensor data from cameras. The cameras can range from microscope cameras to cameras with 360° field of view, and from infrared to range sensors. Computer vision is used in a wide range of applications such as medical image analyses, motion tracking, face recognition, automotive industry, astronomy, and robotics.

One of the advantages of computer vision is that the camera sensor can take pictures of different objects without directly interacting with them. This makes it possible to detect multiple different objects without the need of change the setup. In the scope of this thesis, this means that a camera can detect the position of car parts as well as the specified dimensions and adjustments, regardless of the specifications of the customer. This also means that the computer vision algorithm can adapt to the different CAD models that are provided, and have the flexibility to detect any of these objects within its field of view.

1.1.2 Conformal Geometric Algebra

Conformal geometric algebra is an extension to Euclidean geometry which makes it possible to model geometric objects such as lines, planes and spheres. The algebra is also constructed so that it is possible to calculate the intersections between these objects.

In the scope of this thesis, conformal geometric algebra can be used to describe objects mathematically and to calculate the interactions between them. This makes it possible to model the objects that are detected with computer vision as well as the inverse kinematics of the robots.

1.2 List of publications

This thesis is based on the following publications

Journal publications

[31] A. Kleppe, O. Egeland. Inverse Kinematics for Industrial Robots using Conformal Geometric Algebra. *Modeling, Identification and Control: A Norwegian Research Bulletin*, 37(1):63-75, 2016.

[55] A. Sveier, A. L. Kleppe, L. Tingelstad, O. Egeland. Object Detection in Point Clouds Using Conformal Geometric Algebra. *Advances in Applied Clifford Algebras*, 27(3):1961–1976, 2017.

[30] A. L. Kleppe, A. Bjørkedal, K. Larsen, and O. Egeland. Automated assembly using 3D and 2D cameras. *MDPI Robotics*, 6(3):14, 2017.

[33] A. L. Kleppe, L. Tingelstad, O. Egeland. Coarse Alignment for Model Fitting of Point Clouds using a Curvature-Based Descriptor. *IEEE Trans. Automation Science and Engineering.* Conditionally Accepted.

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

Conference publications

[29] A. Kleppe, L. Tingelstad, and O. Egeland. Initial alignment of point clouds using motors. *ACM International Conference Proceeding Series*, Part F1286, 2017.

1.3 Main Contribution

The objective of this thesis was to investigate solutions that can estimate the position of objects with the accuracy that is required to perform an assembly operation. The solutions produced by this thesis used 3D cameras and conformal geometric algebra to achieve this.

The main contributions of this thesis are

- A proposed analytical inverse kinematics solution for industrial robots using conformal geometric algebra.
- Several proposed methods for object detection and pose estimation, which estimates the position of an object based on its CAD model using a 3D camera. These methods use conformal geometric algebra to achieve this.
- Experimental results which show that the above methods achieves a sub-millimeter accuracy, which is required for assebly.
- Experimental results showing that the same accuracy can be achieved with state-of-the-art pose estimation methods if they are combined with pose estimation methods using a 2D camera.

1.4 Thesis Outline

This thesis is a collection of six papers, and is structured as follows

- **Chapter 2, Point Cloud Alignment** introduces point cloud alignment and the 3D-3D registration problem. It continues to present how point clouds are generated, and how this affects the alignment process.
- **Chapter 3, Descriptors** presents a particular branch of methods that are used for point cloud alignment, called descriptors. The chapter presents the general approach and gives an explanation of some of the state-of-the-art descriptors.
- **Chapter 4, Conformal Geometric Algebra** presents the conformal geometric algebra, which is an extension to the Euclidean algebra. The chapter also explains the different geometric objects that can be generated from it, and how the intersection between them is calculated.

- **Chapter 5, Applications of Conformal Geometric Algebra** presents three applications for conformal geometric algebra, which are used in this thesis: Sphere fitting, model fitting using RANSAC and inverse kinematics.
- **Chapter 6, Curvature-Based Descriptor** presents one of the main contributions of this thesis, the curvature-based descriptor, which uses conformal geometric algebra to perform point cloud alignment.
- **Chapter 7,Experiments** presents the general experimental setup which were used in the different experiments. The experiments themselves are covered in the papers, but this chapter goes more in depth of the overall setup.
- **Chapter 8. Conclusion:** concludes the thesis, and presents some future work.
- **Chapter 9, Publications** presents the published papers.

Chapter 2

Point Cloud Alignment

What do you call a fish with no eyes?

Fsh.

This chapter presents the 3D-3D registration problem and an overview of the types of methods that attempts to solve it. The mechanics of generating point clouds are also presented.

2.1 3D-3D Registration Problem

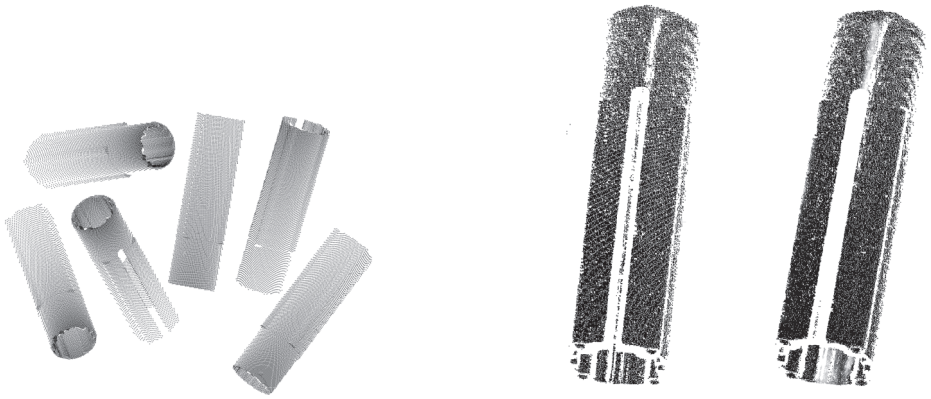
The 3D-3D registration problem [34] is well-established in computer vision, and is still an area of active research. In this problem, the task is to find the optimal displacement between two sets of points. Due to its fundamental nature, it appears in several fields of research, such as in object recognition, tracking, robotics, medical image analysis, graphics and data fusion.

In the 3D-3D registration problem, we define two sets of points, often referred to as *point clouds*: The model point cloud $\boldsymbol{X}$ and the observation point cloud $\boldsymbol{Y}$. These point clouds are usually results from sampling 3D CAD models or scanning objects or scenes with a range sensor, where the model point cloud is the reference object, usually sampled from a CAD model. The observation point cloud is usually captured with a 3D camera, and the task is to find the model point cloud within the scene point cloud. An example of the two types can be seen in Figure 2.1.

The problem involves two sub-problems: Calculation of the displacement between the two point clouds, and estimation of the point correspondences between each point in the point clouds [11].

The mathematical description of the 3D-3D registration problem is: Consider the set of points $\boldsymbol{X} = \{\boldsymbol{x}_i\}$, $i = 1, \ldots, n_x$ with the Euclidean vector representation $\boldsymbol{x}_i \in \mathbb{R}^3$, and the set of points $\boldsymbol{Y} = \{\boldsymbol{y}_j\}$, $j = 1, \ldots, n_y$ with the Euclidean vector representation $\boldsymbol{y}_i \in \mathbb{R}^3$. Then the displacement is calcuated by the minimization of

$$E(\boldsymbol{R}, \boldsymbol{t}) = \sum_{i=1}^{n_x} \|\boldsymbol{y}_{j^*} - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\|^2 \tag{2.1}$$



(a) A set of point clouds of a CAD model

(b) A point cloud of an object taken with a 3D camera

Figure 2.1: A sample of different point clouds taken of the same object; one CAD model and one real object

with respect to $\boldsymbol{R} \in SO(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$, where $\boldsymbol{y}_{j^*}$ is the point in $\boldsymbol{Y}$ which corresponds to the point $\boldsymbol{x}_i$.

The correspondence between point $\boldsymbol{y}_{j^*}$ and $\boldsymbol{x}_i$ is estimated by the minimization problem

$$j^* = \underset{j}{\operatorname{argmin}} \|\boldsymbol{y}_j - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\| \tag{2.2}$$

2.2 Iterative Closest Point (ICP)

A large number of methods have been proposed to solve the registration problem in 3D [51, 11], where ICP [6, 13, 45] is widely used.

The ICP algorithm is an Expectation-Maximization algorithm [34] as it alternates between solving the two sub-problems of displacement and correspondence estimation, until both reaches a local minimum. When solving one of the two sub-problems, the method uses the previous estimate of the displacement sub-problem when solving the point correspondence sub-problem, and vice versa.

The first step of the ICP algorithm is to move both point clouds to the origin. This is done by defining the two point clouds $\widetilde{\boldsymbol{X}}$ and $\widetilde{\boldsymbol{Y}}$ as

$$\widetilde{\boldsymbol{X}} = \{\widetilde{\boldsymbol{x}}_i\} = \{\boldsymbol{x}_i - \overline{\boldsymbol{x}}\}, \qquad \overline{\boldsymbol{x}} = \frac{1}{n_x} \sum_{\boldsymbol{x}_i \in \boldsymbol{X}} \boldsymbol{x}_i, \quad i = 1, \ldots, n_x \tag{2.3}$$

$$\widetilde{\boldsymbol{Y}} = \{\widetilde{\boldsymbol{y}_j}\} = \{\boldsymbol{y}_j - \overline{\boldsymbol{y}}\}, \qquad \overline{\boldsymbol{y}} = \frac{1}{n_y} \sum_{\boldsymbol{y}_j \in \boldsymbol{Y}} \boldsymbol{y}_j, \quad j = 1, \ldots, n_y \tag{2.4}$$

$$\tag{2.5}$$

The translation $\boldsymbol{t}$ is then found by

$$\boldsymbol{t} = \overline{\boldsymbol{y}} - \overline{\boldsymbol{x}} \tag{2.6}$$

After the translation is found, the ICP method finds the rotation matrix and the point correspondence by iteratively solving two optimization problems: The point correspondence and the rotation displacement. First, the point correspondence is found by finding which of the points in $\widetilde{\boldsymbol{Y}}$ are currently closest point to $\widetilde{\boldsymbol{x}}_i$ by solving the equation for each point $\widetilde{\boldsymbol{x}}_i \in \widetilde{\boldsymbol{X}}$

$$j^{*(k)} = \underset{j}{\operatorname{argmin}} \|\widetilde{\boldsymbol{y}}_j^{(k)} - \boldsymbol{R}^{(k)}\widetilde{\boldsymbol{x}}_i\| \tag{2.7}$$

where $k = 1, 2, 3, \ldots n_k$ is the number of iterations of the ICP algorithm. Here, $\boldsymbol{R}^{(1)} = \boldsymbol{I}$.

When the point correspondences are found, the second sub-problem is solved: Finding the displacement between the point clouds. The rotation can be found by solving Procustes' problem

$$\boldsymbol{R} = \underset{\boldsymbol{Q}}{\operatorname{argmin}} \|\boldsymbol{Q}\widetilde{\boldsymbol{x}}_i - \widetilde{\boldsymbol{y}}_j\| \tag{2.8}$$

such that $\boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}} = \boldsymbol{I}$ and $\det \boldsymbol{Q} = 1$.

Procustes' problem is solved by first defining the matrices $[\widetilde{\boldsymbol{X}}]$ and $[\widetilde{\boldsymbol{Y}}]$ as

$$[\widetilde{\boldsymbol{X}}] = \begin{bmatrix} \widetilde{\boldsymbol{x}}_1 & \widetilde{\boldsymbol{x}}_2 & \ldots & \widetilde{\boldsymbol{x}}_{n_x} \end{bmatrix} \tag{2.9}$$

$$[\widetilde{\boldsymbol{Y}}]^{(k)} = \begin{bmatrix} \widetilde{\boldsymbol{y}}_1^{(k)} & \widetilde{\boldsymbol{y}}_2^{(k)} & \ldots & \widetilde{\boldsymbol{y}}_{n_y}^{(k)} \end{bmatrix} \tag{2.10}$$

$$\tag{2.11}$$

and defining the matrix $[\boldsymbol{M}]^{(k)}$ as

$$[\boldsymbol{M}]^{(k)} = [\widetilde{\boldsymbol{Y}}]^{(k)}[\widetilde{\boldsymbol{X}}]^{\mathrm{T}} \tag{2.12}$$

which if solved using singular value decomposition is $[\boldsymbol{M}]^{(k)} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$ and results in the rotation matrix

$$\widetilde{\boldsymbol{R}}^{(k)} = \boldsymbol{U}\mathrm{diag}(1, 1, \det(\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}))\boldsymbol{V}^{\mathrm{T}} \tag{2.13}$$

The point cloud $\widetilde{\boldsymbol{Y}}$ is then updated so that

$$\widetilde{\boldsymbol{y}}_j^{(k+1)} = \widetilde{\boldsymbol{R}}^{(k)}\widetilde{\boldsymbol{y}}_j^{(k)}, \qquad j = 1, 2, 3, \ldots, n_y \tag{2.14}$$

and the rotation matrix is updated so that

$$\boldsymbol{R}^{(k+1)} = \boldsymbol{R}^{(k)}\widetilde{\boldsymbol{R}}^{(k)} \tag{2.15}$$

The final results of the ICP algorithm occurs after k^* iterations, when the method has converged.

A known drawback with Expectation-Maximization algorithms is that they can only achieve local minimal solutions. This means that ICP alone cannot solve the 3D-3D registration problem. In order for ICP to work successfully, two criteria have to be fulfilled:

- *The two point clouds have to be sufficiently close to each other.* This is a results of the Expectation-Maximization algorithm. If the two point clouds are not sufficiently close to each other, the ICP algorithm can potentially be stuck in a local minimum, which is not the global minimum.
- *The two point clouds must overlap each other.* This means that the point clouds has to cover the exact same surface, and that one point cloud must not cover a larger surface than the other. This is because the ICP algorithm is based on solving Procustes problem, which assumes that all the points have a 1-to-1 correspondence. It also only tries to find the correspondence for the points in $\boldsymbol{X}$, which means that the size of $\boldsymbol{X}$ should be smaller than $\boldsymbol{Y}$.

There are several methods that have expanded on the ICP algorithm to make it achieve a global optimal solution, such as GO-ICP [60] and Sparse ICP [9]. Other methods try to find an initial alignment which improves the initial conditions before applying ICP. These methods are called *coarse registration* methods, since they solve the 3D-3D registration problem globally, but not optimally, resulting in a initial pose that converts the 3D-3D registration problem to a local optimization problem.

2.3 Coarse and fine registration

As mentioned, the methods for solving the 3D-3D registration problem can be split into coarse and fine registration methods, where course registration methods are used to calculate the initial conditions for the fine registration methods, such as ICP, which calculates an accurate solution when good initial conditions are given.

Coarse registration problems can be further split into global and local coarse registration problems. Global coarse registration problems use global properties of the point clouds, such as centroid or principal component analysis to find the initial alignment, while local coarse registration methods will match local properties, such as color or curvature in order to find the initial alignment.

The global approaches perform best if the two point clouds overlap. That way the centroid and other global properties are equal in the two point clouds. Examples of these are [38] and [12], where skeletons techniques are used, and [46], where features based on viewpoint is used. [29] is also a global approach which uses the geometric properties of the point cloud to find an initial alignment.

Global approaches have the advantage that the point cloud as a whole is considered. This means that it is invariant to the density of the point cloud, and that the computation time is not necessarily exponentially related to the size of the point cloud. The drawback with the global approaches is that the point clouds have to completely overlap. This has to be taken into consideration when considering the viewpoint of the camera when taking a 3D picture of the object.

The local approaches uses so-called descriptors to produce an initial alignment. A descriptor describes the local properties surrounding a point, such as color, curvature and relationship between surface normals. The descriptors from both point clouds can then be compared to each other, not only relying on the position of the points, but also the shape and curvature of the surrounding surface or the color or texture of the surrounding surface. This results in a more reliable point

correspondence estimate between the descriptors, and therefore the points that they describe. This correspondence is then used to find the displacement between the two point clouds, resulting in an initial alignment. Example of local approach coarse registration methods are: Point Signatures [15], Spin Images [28] and Point Feature Histograms [47, 48].

Local approaches are the most flexible when it comes to partly overlapping point clouds. In ideal situations, a local approach can find the initial alignment between two point clouds that hardly overlap. The drawback with these methods is that they are very dependent on the shape and structure of the point clouds. If a point cloud has few features, such as distinct shapes, edges or corners, then the descriptors are hard to distinguish from each other, making it harder to find the correct correspondence. The descriptors are also dependent on the size of the point cloud, since a descriptor has to be made for each point in the point cloud. In some cases, the descriptors are also dependent on the density of the point cloud, making large and detailed point clouds harder to process.

The use of either global or local approaches to find the initial alignment, depends on the point clouds themselves. If the task is to locate an object within a scene, then the advantage of small overlap goes in favour of the local approaches, while if the point clouds have large featureless surfaces, the global approaches are more suitable, since, in these cases, descriptors are harder to distinguish from each other.

2.4 The Geometry of Point Clouds

The geometry of a point cloud is important to consider when analyzing a point cloud. In this section, some general terms used throughout the thesis are defined, as well as a description of the camera model and its impact on representing geometry.

2.4.1 2D and 3D

The terms 2D and 3D are frequently used in this thesis, and it is important to understand the difference.

2D: are two-dimensional images. These are images where the data points are structured in a two-dimensional grid pattern or pixels. Examples of this are RGB images, where RGB is short for Red, Green, Blue and measurements. In 2D cameras, the measured colors are projected onto an image plane.

3D are three-dimensional images. These images are mostly represented by Euclidean points. Point clouds that sample the whole surface of an object and point clouds that sample the surface of an object viewed from a specific angle are both considered 3D point clouds. 3D images can also be measured with 3D cameras, such as RGB-D cameras, where the D is short for distance.

2.4.2 Surface normals

A surface normal is a vector which is perpendicular to the surface of an object at a given point. This is the vector which defines the tangent plane at the point.

Assume that the surface

$$[f(u,v)] = \begin{bmatrix} f_x(u,v) \\ f_y(u,v) \\ f_z(u,v) \end{bmatrix} \tag{2.16}$$

is a function of u and v and that a given u and v results in the specific Euclidean point $\boldsymbol{p}_{uv} = [f(u,v)]$. Then the surface normal is defined as

$$\boldsymbol{n}_{uv} = \pm \frac{\delta}{\delta u} f(u,v) \times \frac{\delta}{\delta v} f(u,v) \tag{2.17}$$

There are always two surface normals on a given surface, which is represented by the plus and minus. On an enclosed surface, these are often referred to as the inward-pointing normal and outward-pointing normal. It is not possible to determine which is inward-pointing and outward-pointing without any specific information about the surface and the object.

2.5 3D Camera Models in Projective Geometry

When a 3D camera captures an image of a scene, it is processed in the same way as when it is captured with a 2D camera. This section describes how the projective geometry of the camera affects the representation of an object.

2.5.1 Pixel coordinates

The camera frame c is fixed to the camera, where the $x_c y_c$ plane as the focal plane, and the z_c axis is the optical axis pointing out of the camera lens. The origin of c is called the optical center. The relative position between a point $\boldsymbol{p}$ and the optical center is given by the vector $[\boldsymbol{r}]^c_{\boldsymbol{p}}$, such that

$$[\boldsymbol{r}]^c_{\boldsymbol{p}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \qquad [\tilde{\boldsymbol{r}}]^c_{\boldsymbol{p}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.18}$$

where $[\tilde{\boldsymbol{r}}]^c_{\boldsymbol{p}}$ is the homogeneous representation of $[\boldsymbol{r}]^c_{\boldsymbol{p}}$.

All cameras have a camera sensor, which detects the light that enters the camera lens. A camera will typically have in the order of a million small photosensitive light sensors, which detects the incoming light, and each of these corresponds to a pixel. A pixel is the smallest element of a digital image.

When a picture is taken, the distance from the point $\boldsymbol{p}$ and the camera sensor is measured, and also the index of the pixel that measured this distance. The result is a pixel coordinate system, where each pixel contains a distance measurement.

The camera model is also called the pin-hole model, since all the light passes through the optical center of the camera frame, c, and hits the camera sensor which lies on the retinal plane. The retinal plane is parallel to the focal plane, and is placed at a distance f in the negative z_c direction. f is called the focal length.

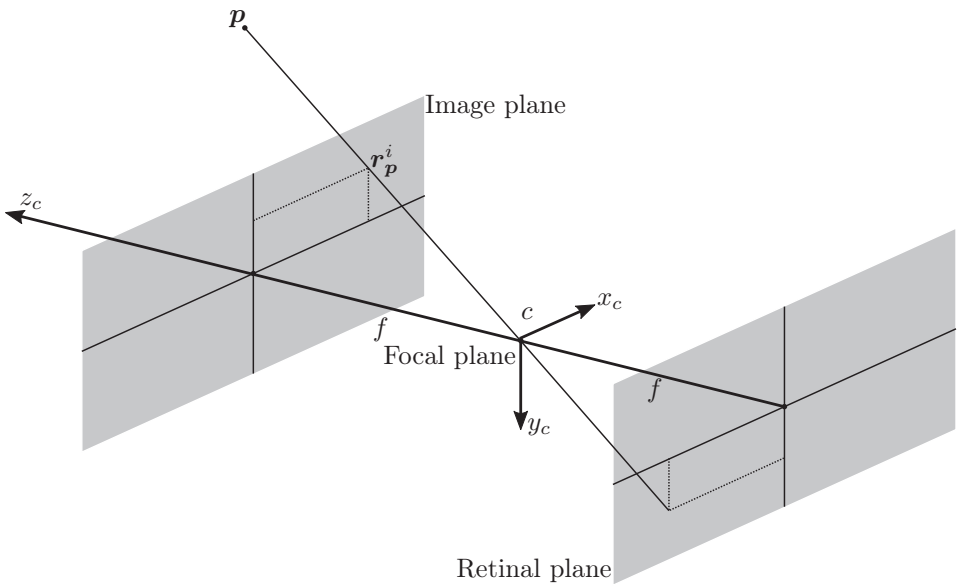


Figure 2.2: The pinhole model. In the picture, focal plane is the $x_c y_c$ plane, and the image plane and retinal plane are parallel to the focal plane with a distance f apart.

The image plane is a virtual plane which is opposed to the physical retinal plane, and is placed a distance f in the positive z_c direction from the focal plane. The image displayed on the image plane is the same as the one captured by the camera sensor on the retinal plane, but rotated by an angle of π around the z_c axis. This is seen in Figure 2.2.

The coordinates of point $\boldsymbol{p}$ when projected onto the image plane. The pixel coordinate vector $[\boldsymbol{r}]^f_{\boldsymbol{p}}$ is therefore

$$[\boldsymbol{r}]^f_{\boldsymbol{p}} = \begin{bmatrix} f\frac{x}{z} \\ f\frac{y}{z} \\ f \end{bmatrix} \tag{2.19}$$

The camera sensor captures one measurement per pixel, and the pixel coordinates are integer coordinates. Given the point $\boldsymbol{p}$, the projection onto the image plane is given as

$$[\tilde{\boldsymbol{p}}]^i = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2.20}$$

where

$$u = \frac{f}{\rho_w}\frac{x}{z} + u_0, \qquad v = \frac{f}{\rho_h}\frac{y}{z} + v_0 \tag{2.21}$$

where ρ_w is the width of a pixel and ρ_h is the height of a pixel. In most cameras $\rho_w = \rho_h$. u_0 and v_0 is the coordinate position of where the z_c axis intersects the image plane.

2.5.2 Distance measurement

In a 3D camera, a distance measurement is taken at each pixel coordinate. This means that when the camera generates a 3D point, the point is given as

$$[\boldsymbol{p}^m] = \begin{bmatrix} \frac{z}{\sqrt{f^2+(u_0-u)^2}}(u_0-u) \\ \frac{z}{\sqrt{f^2+(v_0-v)^2}}(v_0-v) \\ z \end{bmatrix} \tag{2.22}$$

where z is the distance measurement.

It is important to note that the measured point is directly related to z, since the x and y coordinates cannot be measured. This means that any noise that affects the measurement of z will also affect the measurements x and y.

2.5.3 Point Cloud Density

When generating a point cloud with a 3D camera, there are two ways of defining the density of the point cloud: The number of points that are generated by the camera, and the number of points that covers an object in the point cloud.

The number of points that are generated by the camera is directly related to ρ_w and ρ_h. The smaller ρ_w and ρ_h, the greater the point density. This is specified by the resolution of the camera.

The second way of calculating point cloud density is how many points cover an object. This is not only related to the resolution of the camera, but also the distance between the object and the camera, as seen in Figure 2.3. Because of this, the impact of the noise increases the further away the object is from the camera.

2.6 Point Cloud Sampling

Point cloud sampling is when an object is measured and a point cloud is generated to represent that object. This can either be by capturing a point cloud with a 3D camera, or by sampling points from a 3D CAD model.

3D CAD models are generated from a set of points called vertices, and each of them have at least a connection to two other points. The connections are called edges. The smallest geometric shapes that are generated by these connected edges, usually triangles, are called faces.

When sampling a 3D CAD model, the two main methods are overall sampling and viewpoint sampling. Overall sampling generates one point cloud for the whole 3D CAD model. There are several methods for sampling this way, where the most used are:

- **Monte Carlo sampling** [36] also known as uniform random sampling, where n random points on the surface of the CAD model are chosen.

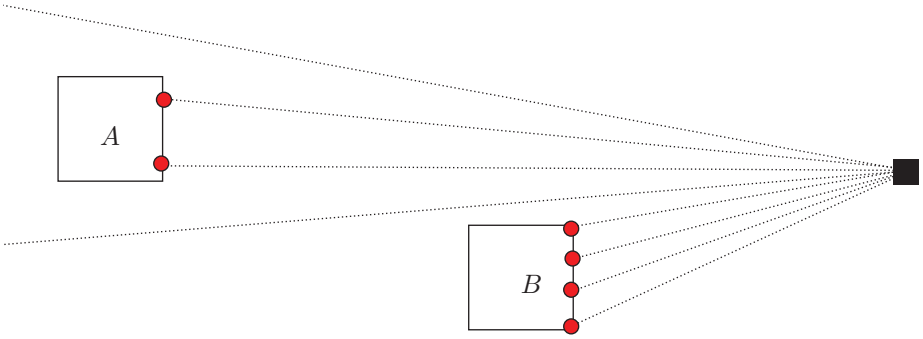


Figure 2.3: The black square represents the 3D camera. The dashed lines are the distance measurements. It can be seen that fewer points are sampled from box A than from box B, because it is further away from the camera.

- **Stratified sampling** [39] or triangle subdivision sampling, where each triangle in the CAD model is sampled based on a voxel grid distributed across the CAD model.
- **Poisson disk sampling** [16] where each vertex in the CAD model is sampled, and then the number of samples are later reduced based on if the local density of the vertices is to large.

The main purpose of these methods is to sample the CAD model as uniformly as possible without any prior knowledge of the model.

With viewpoint sampling, a virtual 3D camera takes pictures of the CAD model from multiple angles [8]. This results in one point cloud per picture taken, as seen in Figure 2.4

The advantage with using multiple viewpoints contrary to sampling the whole point cloud is that if you wish to compare a CAD model to a point cloud captured by a 3D camera, then the scene from the camera represents one viewpoint of the object in question. As seen in Figure 2.5, the points from the CAD model and the scene does not correspond to each other when sampling with overall sampling. Also, if a concave shape is sampled, then the inside is sampled, which will never be seen on a 3D camera, as evident in Figure 2.5.

2.6.1 k-Nearest Neighbour Search vs Radius Search

When using local coarse registration methods, the local properties of a point $\boldsymbol{p}_i$ are analyzed. In order to analyze the local properties, the surface around the given point is measured. Then a set of points $\boldsymbol{N}_i$ is defined which is a neighbourhood of points around point $\boldsymbol{p}_i$. There are mainly two methods of acquiring the set $\boldsymbol{N}_i$: k-nearest neighbour search and radius search.

k-nearest neighbour search (k-NN) [17], is a classification method where the k closest points to $\boldsymbol{p}_i$ are chosen, where k is a user-defined parameter. The method

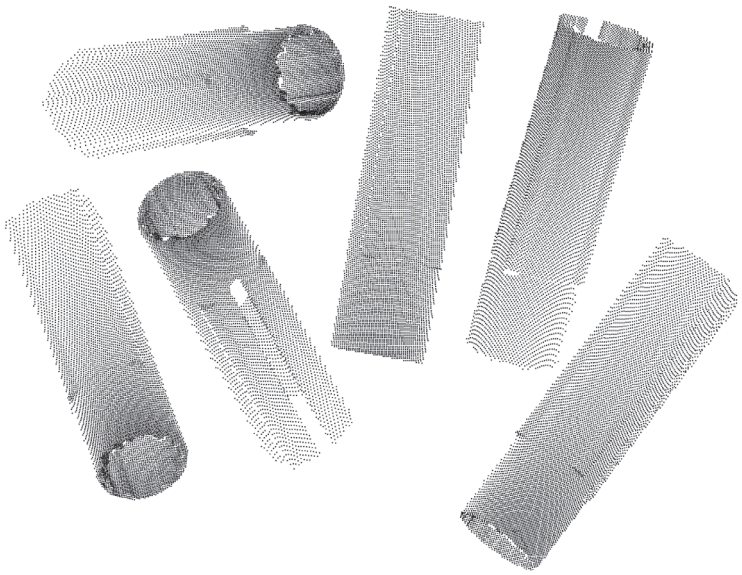


Figure 2.4: Point clouds of one car part from multiple viewpoints.

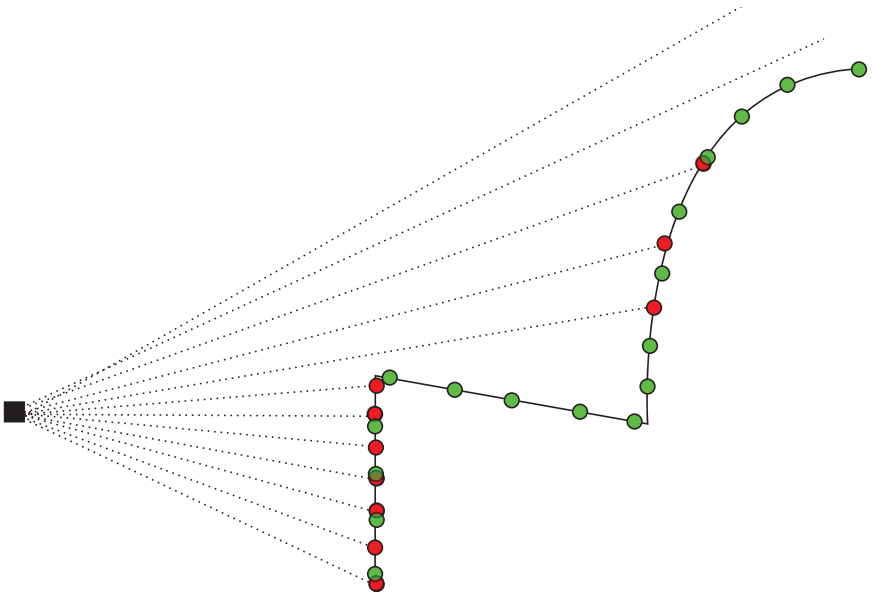


Figure 2.5: The black square represents the 3D camera. The dashed lines are the distance measurements. The red dots are the measurements from a camera or a viewpoint sampling, while the green dot are uniformly sampled such as with overall sampling. It can be seen that there are multiple green dots that do not correspond to any of the red dots captured by the camera.

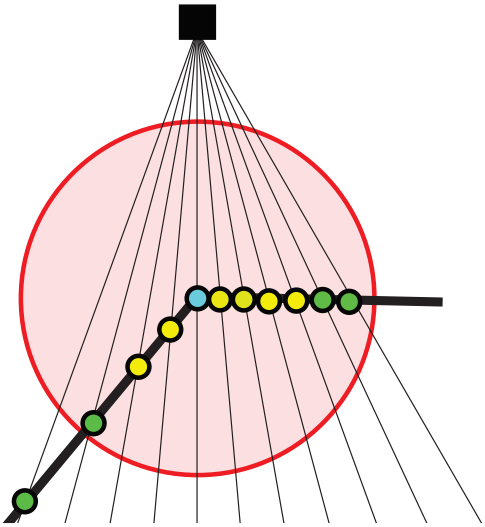


Figure 2.6: Comparison between k-NN and radius search. The blue point is the point $\boldsymbol{p}_i$, and the yellow points are the points selected with a 6-NN search. The points within the red area are selected by the radius search. The black box is the camera.

is mostly used in classification and regression, and is one of the simplest machine learning algorithms.

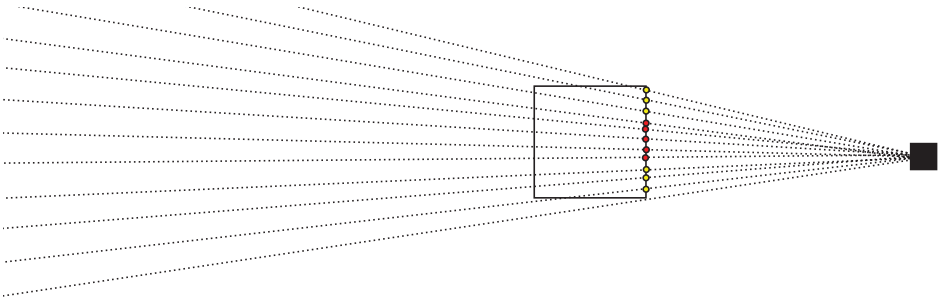
The advantage of using k-NN is that the number of neighbours are consistent, making the computation time consistent and dependent on k. If a point cloud is set up in a k-d tree structure [5], finding the k neighbours is a $\mathrm{O}(k \log n)$ operation, where n is the number of points in the point cloud.

The drawback is that the method is resolution dependent, so that the total surface covered by k points depends on the point cloud density. This also means that when considering the projective geometry that occurs when generating point clouds from a certain angle, the point sampling is skewed, as shown in Figure 2.6. Another problem with the k-nearest neighbour search is that the method is dependent on the distance from the camera. This is because the point density of the point cloud changes the further away the points are, which can be seen in Figure 2.7.

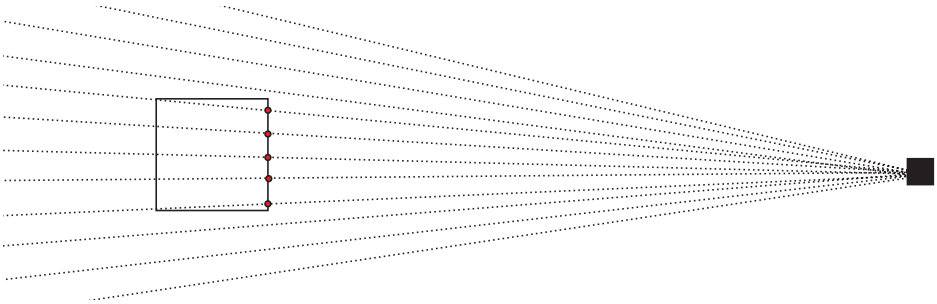
2.6.2 Segmentation

When performing a 3D-3D registration, it is beneficial to use only the parts of the point clouds that are necessary, so that the overlap is as large as possible.

When capturing a point cloud with a 3D camera, the scene usually contains more than just the object that is to be registered, such as tables, floors, walls, backgrounds and even other objects. Segmentation is therefore often used to separate the different parts in the scene, instead of trying to find the object within the whole point cloud.



(a) k-NN performed on an object close to the camera. The result is that about half of the side of the box is selected.



(b) k-NN performed on an object far from the camera. The result is that the whole side of the box is selected.

Figure 2.7: Performing a k-NN with $k = 5$. It can be seen that the result is dependent on the distance between the camera and the object.

This has two effects: It firstly splits one big point cloud into many smaller point clouds. Each point cloud can then be analyzed separately, and can be categorized based on their position, such as backgrounds, or their shape, such as walls and tables which are flat. This limits the number of point clouds and therefore also the search area, which increases the chance of finding a best fit. The second effect is that it makes it possible to perform global coarse registration methods on each point cloud, because the global properties of a point cloud segment will be easier to examine and compare. For instance, if a point cloud depicting objects on a table is segmented into one point cloud per object and one for the table, then the global properties of each point cloud can be analyzed to estimate which object is which.

There are several strategies to perform segmentation [24], which include edge-based segmentation, region growing segmentation, model fitting segmentation and machine learning segmentation:

- **Edge-based segmentation** algorithms [42, 44] detect edges and determine the outlines of a segment, and group together the points that are within the outlines.
- **Region growing segmentation** algorithms [7] start with selecting random points within a point cloud. The algorithm then grows from the points to include their neighbouring points if they have similar characteristics, such as

color and curvature.

- **Model fitting segmentation** algorithms such as Hough Transforms [3] and RANSAC [21] specify a reference model, and segments based on which points fit the model and which does not.
- **Machine learning segmentation** algorithms use classification methods based on machine learning to segment the point cloud.

Which segmentation algorithm yields the best results, depends highly on the type of point cloud. For instance, in [55], RANSAC was chosen, as the majority of the points represented the flat table.

Chapter 3

3D Descriptors

I have a lot of jokes about unemployed people.
But none of them work.

As mentioned in Chapter 2, descriptors are used in coarse registration methods to describe specific object properties of the point cloud, such as colors, textures, shapes and curvatures. Global descriptors describe the point cloud as a whole, and only one descriptor is generated per point cloud. Local descriptors describe the surrounding area around given points, called keypoints, and are therefore generated for each keypoint in the point cloud.

3.1 Keypoints

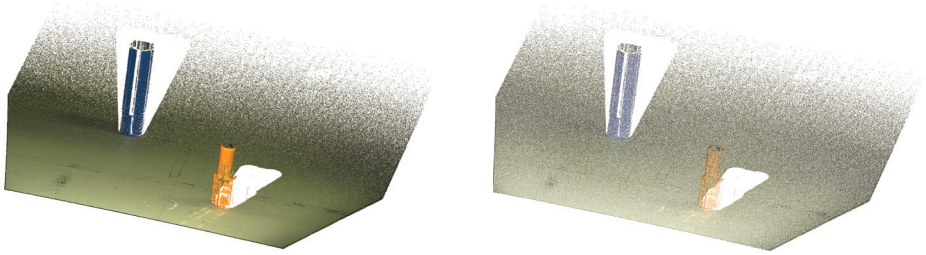
An aspect to consider when calculating descriptors, especially local descriptors, is the size of the point cloud. A point cloud may contain thousands of points, which hampers the execution time of the methods. Another problem with dense point clouds is that the difference between the surrounding area of a point and its neighbouring point may be very small, which makes their descriptors so similar that they are hard to tell apart.

To counter this problem, we can select only a few points in the point cloud, so-called keypoints [50]. This lowers the number of descriptors that has to be generated, and if the selected points are characteristic in some way, the descriptors are also notably different from each other.

For instance, a point cloud depicting a box has large flat surfaces. Most of the points in the point cloud has a surrounding area which is flat, and the points will be difficult to distinguish from each other. In this case, it would be more beneficial to select the points near the corner of the box, since these have a more unique characteristics. An example of a keypoint extraction is seen in Figure 3.1.

There are several methods to select keypoints, and they are mostly dependent on prior knowledge of the point cloud. In the experiments done in the thesis, the prior knowledge is which objects are being detected, the general shape of these objects, and that these objects are placed on a table. It is possible to use this information to select the keypoints.

Figure 3.1: An example of a keypoint extraction method. This image is taken from [23]



(a) Original point cloud containing 2 304 000 points

(b) Point cloud after down-sampling with a voxel size of 1 mm, containing 375 936 points

Figure 3.2: Down-sampling using voxel grids

3.1.1 Down-sampling

A different approach to lowering the execution time when calculating descriptors is down-sampling or sub-sampling. Here, the number of points is reduced to a fixed number, without taking any prior knowledge into account. A widely used approach is to create a three dimensional grid, also called voxels, which is the three-dimensional version of pixels. All points within a voxel are grouped together and averaged into one aggregate point. The effect of this is to reduce the number of points and to grant a consistent density throughout the point cloud. This is used is in [46, 47], and a down-sampled point cloud can be seen in Figure 3.2.

A potential problem with down-sampling is that the number of points is reduced, which reduces the information content of the point cloud. Moreover, the resulting density of the point cloud will be lower than the lowest density of the original point cloud. This limits the accuracy of the representation of the real object, and therefore also the accuracy of the descriptors that are based on the down-sampled point cloud.

3.2 Principal Component Analysis (PCA)

Principal Component Analysis is widely used in statistics, and is used to find the orthogonal frame where the given data set consists of linearly uncorrelated values, also called *components* [56].

The method is based on the computation of a covariance matrix $\mathbf{C} \in \mathbb{R}^{3\times3}$ defined by

$$\mathbf{C} = \sum_{\boldsymbol{p}_i \in \boldsymbol{X}} ([\boldsymbol{p}_i] - \bar{\boldsymbol{p}})([\boldsymbol{p}_i] - \bar{\boldsymbol{p}})^{\mathrm{T}} \tag{3.1}$$

where $\bar{\boldsymbol{p}} = \frac{1}{|\boldsymbol{X}|} \sum_{\boldsymbol{p}_i \in \boldsymbol{X}} \boldsymbol{p}_i$ is the centroid of all the points in the point cloud $\boldsymbol{X}$, and $|\boldsymbol{X}|$ is the number of points in $\boldsymbol{X}$.

The covariance matrix $\mathbf{C}$ can be decomposed as

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i, \qquad i = 1, 2, 3 \tag{3.2}$$

to find the eigenvalues λ_i, and their corresponding eigenvectors $\mathbf{v}_i$.

Let the eigenvalues be ordered so that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$. Then the direction of $\mathbf{v}_1$ is the direction where the data has the most variation, and $\mathbf{v}_3$ is the direction with the least variation in the data.

PCA can be used to generate a local reference frame [56], making it possible to create local descriptors which are consistent. A known issue with using PCA for this, is that the directions of $\mathbf{v}_1$, $\mathbf{v}_2$ and therefore also $\mathbf{v}_3$ have two solutions. There are several solutions that try to solve this problem, such as [10], where the sign of the eigenvector $\mathbf{v}_i$ is determined by calculating the direction of where the majority of the data is represented. This is done by calculating the equation

$$s_j = \sum_{\in \boldsymbol{X}} \mathrm{sign}(\mathbf{v}_j^{\mathrm{T}} \boldsymbol{p}_i)(\mathbf{v}_j^{\mathrm{T}} \boldsymbol{p}_i)^2 \tag{3.3}$$

where $\boldsymbol{p}_i \in \boldsymbol{X}$, and s_j is the non-normalized sign of $\mathbf{v}_j$.

3.3 Normal Estimation

3D CAD models will have a defined surface for each face of the model, which makes it possible to include the surface normal when generating point clouds from 3D models.

When capturing a 3D image with a camera, however, the surface normal is not defined, and an estimate is required. As mentioned in Section 2.4.2, the surface normal is the normal to the tangent plane at a point. There are several methods to estimate the surface normal, and a frequently used method is to find the normal of the least-squares estimate of the plane [37]. The least-squares solution is estimated at the given point based on the surrounding points. Principal component analysis can be used to calculate the least-squares plane estimate, where the plane estimate is spanned from $\mathbf{v}_1$ and $\mathbf{v}_2$. The surface normal of the plane estimate is therefore given as $\pm\mathbf{v}_3$.

Since every surface has two sides, there will be two possible surface normals at any given point. On a point cloud captured by a 3D camera it is possible to know which surface normal is correct, since there is only one side of the surface that is visible. To calculate which of $\mathbf{v}_3$ and $-\mathbf{v}_3$ is the visible surface normal, the angle between the camera and the surface normal is calculated, and normal is selected from:

$$\boldsymbol{n}_i = \mathbf{v}_3, \qquad \text{if } \mathbf{v}_3 \cdot \boldsymbol{p}_i < 0 \tag{3.4}$$

$$\boldsymbol{n}_i = -\mathbf{v}_3, \qquad \text{if } \mathbf{v}_3 \cdot \boldsymbol{p}_i \geq 0 \tag{3.5}$$

$$\tag{3.6}$$

where $\boldsymbol{n}_i$ is the surface normal at $\boldsymbol{p}_i$, and $\mathbf{v}_3$ is calculated using PCA of the neighbouring points, $\boldsymbol{N}_i$, of $\boldsymbol{p}_i$. It is assumed that the camera is at the origin.

3.4 Spherical Coordinate System

There are several descriptor methods that use spherical coordinates rather than Cartesian coordinates. This is because in a local reference frame of a point, it is easier to segment the relative position between the center point and the surrounding points using spherical coordinates.

A spherical coordinate system is constructed by selecting two orthogonal directions, called the zenith, $\boldsymbol{z}$, and the azimuth reference, $\boldsymbol{x}$; and the origin, $\boldsymbol{o}$. A reference plane is generated which intersects the origin and where the zenith axis is the normal. The azimuth is a reference direction in the plane. The spherical coordinates of a point $\boldsymbol{p}$ is given by

Radius r**:** The Euclidean distance between the point $\boldsymbol{p}$ and the origin $\boldsymbol{o}$.

Inclination ϕ**:** The angle between the zenith $\boldsymbol{z}$ and line segment $\boldsymbol{op}$, constructed by $\boldsymbol{o}$ and $\boldsymbol{p}$. This is also called the polar angle.

Azimuth θ**:** The angle between the azimuth reference direction $\boldsymbol{x}$ and the projection of the line segment $\boldsymbol{op}$ onto the reference plane.

Figure 3.3 shows the setup of the spherical coordinate system.

In the descriptors below, the local coordinate system is generated by selecting a reference point as the origin, and the normal of that point as the zenith. The methods use different techniques for selecting the azimuth reference.

The spherical coordinates are used for constructing histograms, where the spherical coordinates are divided into smaller subsections, or so called bins. The number of bins and how they are divided depends on the descriptor. These subdivisions are referenced as radial, azimuth and elevation subdivisions, where radial divides r, azimuth divides θ and the elevation divides ϕ.

3.5 State-of-the-Art Descriptors

This section lists four descriptors that have very different approaches to generating descriptors. They are widely used in 3D point cloud alignment.

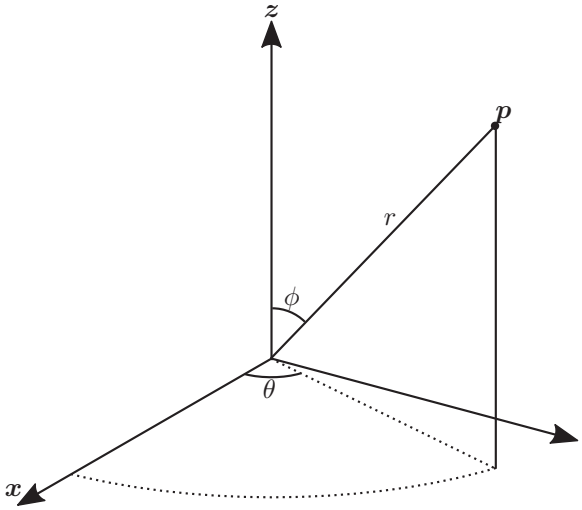


Figure 3.3: An illustration of the spherical coordinate system

3.5.1 Point Feature Histogram

The Point Feature Histogram (PFH) [47] generates a histogram for each point in the point cloud. The method inputs the neighbouring points of that point and calculates the relationship between them to define the histogram. When histograms are generated, they are compared with the histograms from other point clouds to find the point correspondences.

The algorithm for the PFH method is: For each point $\boldsymbol{p}_i$ find the k-nearest neighbour points, generating the set $\boldsymbol{N}_i$. For each point pair $\boldsymbol{p}_j, \boldsymbol{p}_k \in \boldsymbol{N}_i$, we define $\boldsymbol{p}_s$ and $\boldsymbol{p}_t$ as

$$\boldsymbol{p}_s = \boldsymbol{p}_j, \quad \boldsymbol{p}_t = \boldsymbol{p}_k, \qquad \text{if } \boldsymbol{n}_j \cdot (\boldsymbol{p}_j - \boldsymbol{p}_k) \leq \boldsymbol{n}_k \cdot (\boldsymbol{p}_j - \boldsymbol{p}_k) \tag{3.7}$$

$$\boldsymbol{p}_s = \boldsymbol{p}_k, \quad \boldsymbol{p}_t = \boldsymbol{p}_j, \qquad \text{if } \boldsymbol{n}_j \cdot (\boldsymbol{p}_j - \boldsymbol{p}_k) > \boldsymbol{n}_k \cdot (\boldsymbol{p}_j - \boldsymbol{p}_k) \tag{3.8}$$

where $\boldsymbol{n}_j$ and $\boldsymbol{n}_k$ are the normals of $\boldsymbol{p}_j$ and $\boldsymbol{p}_k$ respectively. When $\boldsymbol{p}_s$ and $\boldsymbol{p}_t$ is defined, a Darboux frame is constructed where

$$\boldsymbol{u} = \boldsymbol{n}_s \tag{3.9}$$

$$\boldsymbol{v} = \frac{(\boldsymbol{p}_t - \boldsymbol{p}_s) \times \boldsymbol{u}}{||\boldsymbol{p}_t - \boldsymbol{p}_s||} \tag{3.10}$$

$$\boldsymbol{w} = \boldsymbol{u} \times \boldsymbol{v} \tag{3.11}$$

This is then further used to to calculate 4 features

$$f_1 = \boldsymbol{v} \cdot \boldsymbol{n}_t \tag{3.12}$$
$$f_2 = ||\boldsymbol{p}_t - \boldsymbol{p}_s|| \tag{3.13}$$
$$f_3 = \frac{\boldsymbol{u} \cdot (\boldsymbol{p}_t - \boldsymbol{p}_s)}{f_2} \tag{3.14}$$
$$f_4 = \text{atan2}(\boldsymbol{w} \cdot \boldsymbol{n}_t, \boldsymbol{u} \cdot \boldsymbol{n}_t) \tag{3.15}$$

which is used to generate the index i_x

$$i_x = \sum_{i=1}^{4} \left\lfloor \frac{f_i \cdot d}{f_i \text{imax} - f_{i\text{min}}} \right\rfloor \cdot d^{i-1} \tag{3.16}$$

where $f_{i\text{min}}$ and $f_{i\text{max}}$ is the theoretical smallest and largest values of f_i, and d is the subdivision of the range $(f_{i\text{max}} - f_{i\text{min}})$.

The formula above helps to generate an index. The index has 16 categories, each representing the 2^4 possibilities that the four features can generate. The first category is $f_1 \geq 0$, $f_2 \geq r$, $f_3 \geq 0$ and $f_4 \geq 0$, while the second category is $f_1 < 0$, $f_2 \geq r$, $f_3 \geq 0$ and $f_4 \geq 0$, and so on for every combination of f_i. Here, r is a user-defined radius.

While summing up all these indices for all the possible point pairs in $\boldsymbol{N}_i$, a histogram is generated, which characterizes the point $\boldsymbol{p}_i$. This histogram is then compared to other histograms to find the point correspondences.

Fast Point Feature Histogram

A similar approach to the PFH is the Fast PFH [48]. In [48], they first define the Simplified Point Feature Histogram (SPFH), which uses the same principals as PFH, but does not use all the point pair combinations in $\boldsymbol{N}_i$, only the ones including $\boldsymbol{p}_i$. The method also reduces the number of features from four to three, omitting f_2. The FPFH uses the SPFH and includes an influence region

$$\text{FPFH}(\boldsymbol{p}_i) = \text{SPFH}(\boldsymbol{p}_i) + \frac{1}{k} \sum_{\boldsymbol{p}_k \in \boldsymbol{N}_i} \frac{1}{\omega_k} \text{SPFH}(\boldsymbol{p}_k) \tag{3.17}$$

where ω_k is the distance between $\boldsymbol{p}_i$ and $\boldsymbol{p}_k$. The method further optimizes the histogram representation, and represents the histogram in 3 histograms, one for each feature.

3.5.2 Point Pair Features

The Point Pair Features (PPF) [20] descriptor generates a feature $\boldsymbol{F}_{ij}$ for each point pair $(\boldsymbol{p}_i, \boldsymbol{p}_j) \in \boldsymbol{X}$ given by

$$\boldsymbol{F}_{ij} = (||\boldsymbol{d}||, \angle(\boldsymbol{n}_i, \boldsymbol{d}), \angle(\boldsymbol{n}_j, \boldsymbol{d}), \angle(\boldsymbol{n}_i, \boldsymbol{n}_j)) \tag{3.18}$$

where $\boldsymbol{d} = \boldsymbol{p}_i - \boldsymbol{p}_j$, and $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ are the normals for $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ respectively.

Voting Scheme

Each of the features $\boldsymbol{F}_{ij}$ are placed in a hash table, both for the model point cloud $\boldsymbol{M}$ and the scene point cloud $\boldsymbol{S}$. For each point pair in $\boldsymbol{S}$, the features $\boldsymbol{F}_{\boldsymbol{S}ij}$ are looked up in the hash table. Then the method searches for a set of similar features $\boldsymbol{F}_{\boldsymbol{M}ij}$ from $\boldsymbol{M}$. Each of the features gets a vote, and the displacement between $\boldsymbol{F}_{\boldsymbol{S}ij}$, and $\boldsymbol{F}_{\boldsymbol{M}ij}$ is calculated. This continues for all point pairs in $\boldsymbol{S}$. After this, the feature $\boldsymbol{F}_{\boldsymbol{M}ij}$ with the most votes is chosen, and the displacement between $\boldsymbol{F}_{\boldsymbol{M}ij}$ and $\boldsymbol{F}_{\boldsymbol{S}ij}$ is given as the pose estimation.

3.5.3 Signature of Histogram of OrienTation

Signature of Histogram of OrientTation (SHOT) [56], uses both signatures and histograms, based on the argument that signatures generate 3D structures based on neighbouring points, while histograms uses neighbouring points to generate geometric topologies.

The method first generates a reference frame using PCA, with a slight alteration using [10] so that the frame is not flipped around when there are small changes in the point positions. The next step is to define a ball around each point $\boldsymbol{p}_i$. This ball is partitioned in 32 volumes or bins by discretizing the azimuth into 8 values, while the elevation into 2 values and the radius into 2 values. This gives the $8 \cdot 2 \cdot 2 = 32$ bins. The neighbouring points $\boldsymbol{N}_i$ of $\boldsymbol{p}_i$ are then categorized into each bin based on their position relative to the point $\boldsymbol{p}_i$. For each bin, a histogram is generated by calculating the dot product, $h_{ij} = \boldsymbol{n}_i \cdot \boldsymbol{n}_j$, between the normal of $\boldsymbol{p}_i$ and the normal for each point in the bin $\boldsymbol{p}_j$.

When the histograms are calculated, they are grouped into one descriptor.

Because the reference frame is calculated as mentioned in [10], the descriptor is considered rotation invariant.

3.5.4 Radial-based Surface Descriptor

The Radial-based Surface Descriptor (RSD) [35] describes the radial relationship between a point and its neighbourhood. The algorithm establishes a neighbourhood $\boldsymbol{N}_i$ for each point $\boldsymbol{p}_i$. For each point $\boldsymbol{p}_j \in \boldsymbol{N}_i$, the algorithm generates a sphere where $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ is on the surface, and the surface normals $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ are considered surface normals on the sphere. The features

$$d_{ij} = |\boldsymbol{p}_i - \boldsymbol{p}_j| \tag{3.19}$$

$$\cos \alpha_{ij} = \boldsymbol{n}_i \cdot \boldsymbol{n}_j \tag{3.20}$$

$$r_{ij}^2 = \frac{d_{ij}^2}{2(1 - \cos \alpha_{ij})} \tag{3.21}$$

are calculated for each point pair $(\boldsymbol{p}_i, \boldsymbol{p}_j)$. When this is done for all points in $\boldsymbol{N}_i$, the maximum and minimum radius is used to define the descriptor, and the relationship between the two defines the surrounding surface area. If $r_{\max} \approx r_{\min}$, then the form is spherical or planar, and the bigger difference between $r_{\max}$ and $r_{\min}$, the surrounding area has a cylindrical form. This descriptor is then used to find the point correspondences between the model and data point clouds.

3.5.5 3D Shape Context

3D Shape Context (3DSC) [22] is an extension of the 2D version, 2DSC, presented in [4]. 3DSC descriptors are generated for each point $\boldsymbol{p}$, and uses spherical coordinates to place neighbouring points $\boldsymbol{N}_i$ in bins. The surface normal defines where the azimuth is zero. The points are placed in bins, where the division of the azimuth and elevation is equal, while the radial division is logarithmic, making smaller divisions closer to the point $\boldsymbol{p}$. A minimum radius is defined, preventing very small bins, which are very subjected to noise.

Since the descriptor has a degree of freedom in the azimuth direction, it is not possible to know the rotation of the descriptor. To prevent this, the descriptor is rotated n times, where n is the number of divisions in the azimuth, which results in n descriptors per point.

Chapter 4

Conformal Geometric Algebra

Why did the farmer get an award?
Because he was out standing in his field.

This chapter presents an introduction to conformal geometric algebra [19, 26]. Conformal geometric algebra is an extension of Euclidean geometric methods, and allows for an efficient description of spheres, planes, lines and points, including the displacement of such objects. Moreover, conformal geometric algebra is a valuable tool for calculating the intersection between geometric objects and distances between them.

In this thesis, the underlying physical space is the three-dimensional Euclidean space, which is represented in conformal geometric algebra by a five-dimensional space.

4.1 Euclidean Space

The basis of the Euclidean space $\mathbb{R}^3$ is given by the orthonormal vectors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$. A vector $\boldsymbol{a}$ is represented by

$$\boldsymbol{a} = a_1\boldsymbol{e}_1 + a_2\boldsymbol{e}_2 + a_3\boldsymbol{e}_3 \tag{4.1}$$

where $a_i = \boldsymbol{a} \cdot \boldsymbol{e}_i$ is the coordinate of $\boldsymbol{a}$ with respect to the basis.

We define $\boldsymbol{b}$ and $\boldsymbol{c}$ in the same manner

$$\boldsymbol{b} = b_1\boldsymbol{e}_1 + b_2\boldsymbol{e}_2 + b_3\boldsymbol{e}_3 \qquad \boldsymbol{c} = c_1\boldsymbol{e}_1 + c_2\boldsymbol{e}_2 + c_3\boldsymbol{e}_3 \tag{4.2}$$

The vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ are used throughout this section to define various properties of the Euclidean space.

4.1.1 Inner Product

The inner product between the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as

$$\begin{aligned} \boldsymbol{a} \cdot \boldsymbol{b} &= (a_1\boldsymbol{e}_1 + a_2\boldsymbol{e}_2 + a_3\boldsymbol{e}_3) \cdot (b_1\boldsymbol{e}_1 + b_2\boldsymbol{e}_2 + b_3\boldsymbol{e}_3) \\ &= a_1b_1 + a_2b_2 + a_3b_3 \end{aligned} \tag{4.3}$$

where

$$\boldsymbol{e}_i \cdot \boldsymbol{e}_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{4.4}$$

It is noted that the inner product er commutative, so that

$$\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{b} \cdot \boldsymbol{a} \tag{4.5}$$

This means that the length $|\boldsymbol{a}|$ of $\boldsymbol{a}$ is given by

$$\boldsymbol{a} \cdot \boldsymbol{a} = |\boldsymbol{a}|^2 \tag{4.6}$$

Some other useful properties of the inner product are

$$\alpha \cdot \boldsymbol{a} = \alpha \boldsymbol{a} \tag{4.7}$$
$$(\boldsymbol{a} \cdot \boldsymbol{b}) \cdot \boldsymbol{c} = \boldsymbol{a} \cdot (\boldsymbol{b} \cdot \boldsymbol{c}) \tag{4.8}$$
$$(\boldsymbol{a} + \boldsymbol{b}) \cdot \boldsymbol{c} = \boldsymbol{a} \cdot \boldsymbol{c} + \boldsymbol{b} \cdot \boldsymbol{c} \tag{4.9}$$

where $\alpha \in \mathbb{R}$ is a scalar.

4.1.2 Outer Product

The outer product $\boldsymbol{e}_i \wedge \boldsymbol{e}_j$ between two orthonormal vectors $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ satisfies

$$\boldsymbol{e}_i \wedge \boldsymbol{e}_j = -\boldsymbol{e}_j \wedge \boldsymbol{e}_i \tag{4.10}$$

which implies $\boldsymbol{e}_i \wedge \boldsymbol{e}_i = 0$.

The outer product between the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is given by

$$\begin{aligned} \boldsymbol{a} \wedge \boldsymbol{b} &= (a_1\boldsymbol{e}_1 + a_2\boldsymbol{e}_2 + a_3\boldsymbol{e}_3) \wedge (b_1\boldsymbol{e}_1 + b_2\boldsymbol{e}_2 + b_3\boldsymbol{e}_3) \\ &= (a_2b_3 - a_3b_2)\boldsymbol{e}_{23} + (a_3b_1 - a_1b_3)\boldsymbol{e}_{31} + (a_1b_2 - a_2b_1)\boldsymbol{e}_{12} \end{aligned} \tag{4.11}$$

Since the outer product is anti-commutative, then

$$\boldsymbol{a} \wedge \boldsymbol{b} = -\boldsymbol{b} \wedge \boldsymbol{a} \tag{4.12}$$

It also follows that

$$\boldsymbol{a} \wedge \boldsymbol{a} = 0 \tag{4.13}$$

It is noted that the cross product has the same coefficients as the outer product, so that

$$\boldsymbol{a} \times \boldsymbol{b} = (a_2b_3 - a_3b_2)\boldsymbol{e}_1 + (a_3b_1 - a_1b_3)\boldsymbol{e}_2 + (a_1b_2 - a_2b_1)\boldsymbol{e}_3 \tag{4.14}$$

Note that the basis of $\boldsymbol{a} \wedge \boldsymbol{b}$ is $\{\boldsymbol{e}_{23}, \boldsymbol{e}_{31}, \boldsymbol{e}_{12}\}$, while the basis of $\boldsymbol{a} \times \boldsymbol{b}$ is $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$. In the terminology of geometric algebra, $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$ is the dual basis of $\{\boldsymbol{e}_{23}, \boldsymbol{e}_{31}, \boldsymbol{e}_{12}\}$ in Euclidean space.

The cross product is in fact the dual form of the outer product. Duality is described later in this section.

Some useful properties of the outer product are

$$\alpha \wedge \boldsymbol{a} = \alpha \boldsymbol{a} \tag{4.15}$$
$$(\boldsymbol{a} \wedge \boldsymbol{b}) \wedge \boldsymbol{c} = \boldsymbol{a} \wedge (\boldsymbol{b} \wedge \boldsymbol{c}) \tag{4.16}$$
$$(\boldsymbol{a} + \boldsymbol{b}) \wedge \boldsymbol{c} = \boldsymbol{a} \wedge \boldsymbol{c} + \boldsymbol{b} \wedge \boldsymbol{c} \tag{4.17}$$

where $\alpha \in \mathbb{R}$.

4.1.3 Geometric Product

The geometric product is defined as the sum of the inner and outer product

$$\boldsymbol{ab} = \boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \wedge \boldsymbol{b} \tag{4.18}$$

This means that

$$\boldsymbol{aa} = \boldsymbol{a}^2 = \boldsymbol{a} \cdot \boldsymbol{a} + \boldsymbol{a} \wedge \boldsymbol{a} = |\boldsymbol{a}|^2 \tag{4.19}$$

and

$$\boldsymbol{ba} = \boldsymbol{b} \cdot \boldsymbol{a} + \boldsymbol{b} \wedge \boldsymbol{a} = \boldsymbol{a} \cdot \boldsymbol{b} - \boldsymbol{a} \wedge \boldsymbol{b} \tag{4.20}$$

Because of (4.18) and (4.20), the inner and outer product of vectors can also be defined by the geometric product where

$$\boldsymbol{a} \cdot \boldsymbol{b} = \frac{1}{2}(\boldsymbol{ab} + \boldsymbol{ba}) \tag{4.21}$$
$$\boldsymbol{a} \wedge \boldsymbol{b} = \frac{1}{2}(\boldsymbol{ab} - \boldsymbol{ba}) \tag{4.22}$$

4.1.4 Blades

A vector $\boldsymbol{a} \in \mathbb{R}^3$ is said to be a blade of grade 1, as the space $\mathbb{R}^3$ is spanned from the orthogonal vectors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$. The outer product can generate higher grades between vectors, and a k-blade is defined as the outer product between k vectors. For instance, $\boldsymbol{a} \wedge \boldsymbol{b}$ is an example of a 2-blade, which is of grade 2. The basis for grade 2 blades in Euclidean space is

$$\{\boldsymbol{e}_{23}, \boldsymbol{e}_{31}, \boldsymbol{e}_{12}\} \tag{4.23}$$

Another property of blades is that

$$\boldsymbol{a}_1 \wedge \boldsymbol{a}_2 \wedge \cdots \wedge \boldsymbol{a}_k = 0, \qquad k > n, \boldsymbol{a}_i \in \mathbb{R}^n \tag{4.24}$$

which means that the highest grade achieved in $\mathbb{R}^3$ is 3.

A 3-blade in Euclidean $\mathbb{R}^3$ can be written as

$$\begin{aligned}\boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c} &= (\boldsymbol{a}\wedge\boldsymbol{b})\wedge\boldsymbol{c}\\ &= ((a_2b_3-a_3b_2)\boldsymbol{e}_{23}+(a_3b_1-a_1b_3)\boldsymbol{e}_{31}+(a_1b_2-a_2b_1)\boldsymbol{e}_{12})\\ &\wedge(c_1\boldsymbol{e}_1+c_2\boldsymbol{e}_2+c_3\boldsymbol{e}_3)\\ &= \alpha(e_1\wedge\boldsymbol{e}_2\wedge\boldsymbol{e}_3)=\alpha\boldsymbol{e}_{123}\end{aligned} \tag{4.25}$$

where α is an arithmetical expression of a_i, b_i and c_i for $i=1,2,3$. The basis for grade 3 blades in Euclidean space is

$$\{\boldsymbol{e}_{123}\} \tag{4.26}$$

This in turn means that the basis of the multivectors of $\mathbb{R}^3$ is given by

$$\{1,\boldsymbol{e}_1,\boldsymbol{e}_2,\boldsymbol{e}_3,\boldsymbol{e}_{23},\boldsymbol{e}_{31},\boldsymbol{e}_{12},\boldsymbol{e}_{123}\} \tag{4.27}$$

An example of a 2-blade and 3-blade is given in Figure 4.1.

4.1.5 Multivectors

Multivectors are the sums of different k-blades, where a k-vector is defined as the sum of k-blades. For instance, a bivector is defined as

$$\boldsymbol{A}_{\langle 2\rangle}=a_{12}\boldsymbol{e}_{12}+a_{23}\boldsymbol{e}_{23}+a_{31}\boldsymbol{e}_{31} \tag{4.28}$$

A multivector can also be the sum of different types of blades, and the most general form is

$$\boldsymbol{A}=a_1\boldsymbol{e}_1+a_2\boldsymbol{e}_2+a_3\boldsymbol{e}_3+a_{12}\boldsymbol{e}_{12}+a_{23}\boldsymbol{e}_{23}+a_{31}\boldsymbol{e}_{31}+a_{123}\boldsymbol{e}_{123} \tag{4.29}$$

which includes all the base elements of the Euclidean space.

Reverse and Inverse of Blades

The reverse of a blade $\boldsymbol{A}_{\langle n\rangle}$ is denoted $\widetilde{\boldsymbol{A}}_{\langle n\rangle}$, and is given by

$$\boldsymbol{A}_{\langle n\rangle}=\boldsymbol{e}_1\wedge\boldsymbol{e}_2\wedge\cdots\wedge\boldsymbol{e}_n \tag{4.30}$$

$$\widetilde{\boldsymbol{A}}_{\langle n\rangle}=\boldsymbol{e}_n\wedge\boldsymbol{e}_{n-1}\wedge\cdots\wedge\boldsymbol{e}_1 \tag{4.31}$$

Since the outer product has the property given in (4.12), the reverse of a blade $\boldsymbol{A}_{\langle k\rangle}$ can be written as

$$\tilde{\boldsymbol{A}}_{\langle k\rangle}=(-1)^{k(k-1)/2}\boldsymbol{A}_{\langle k\rangle} \tag{4.32}$$

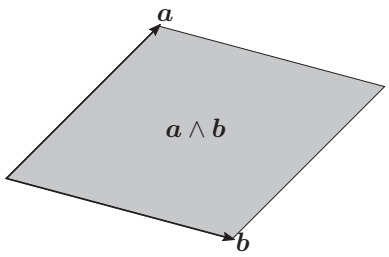
For the three-dimensional Euclidean space, this means that

$$\widetilde{\boldsymbol{A}}=\boldsymbol{a}\wedge\boldsymbol{b}\wedge\boldsymbol{c}=\boldsymbol{c}\wedge\boldsymbol{b}\wedge\boldsymbol{a}=-\boldsymbol{A} \tag{4.33}$$

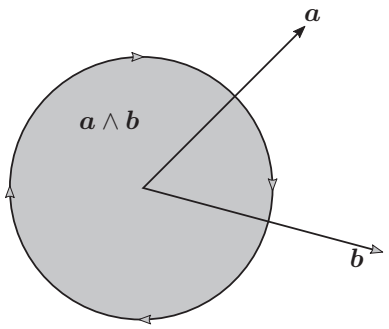
The inverse of a blade $\boldsymbol{A}$ is denoted $\boldsymbol{A}^{-1}$, and is defined as

$$\boldsymbol{A}^{-1}=\frac{\widetilde{\boldsymbol{A}}}{|\boldsymbol{A}|^2} \tag{4.34}$$

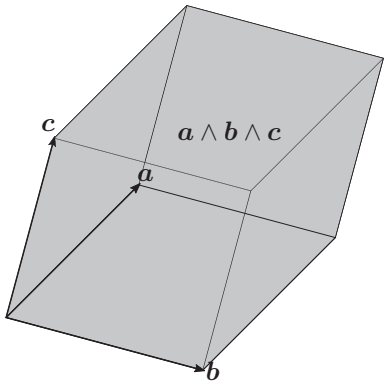
and is therefore the same as the reverse of the blade if $|\boldsymbol{A}|^2=1$.



(a) A 2-blade representation. Here, the size of $\boldsymbol{a} \wedge \boldsymbol{b}$ is the area of the parallelogram.



(b) A 2-blade representation. Here, the size of $\boldsymbol{a} \wedge \boldsymbol{b}$ is the area of the circle. The orientation of the blade is indicated by the arrows



(c) A 3-blade representation. Here, the size of $\boldsymbol{a} \wedge \boldsymbol{b} \wedge \boldsymbol{c}$ is the area of the volume.

Figure 4.1: Two different representations of a 2-blade, and a representation of the 3-blade.

4.1.6 Pseudoscalar and Duals

The pseudoscalar is defined as the blade of highest grade in the specified space. In a Euclidean space $\mathbb{R}^3$, the pseudoscalar is defined as $\boldsymbol{I}_E = \boldsymbol{e}_{123} = \boldsymbol{e}_1 \wedge \boldsymbol{e}_2 \wedge \boldsymbol{e}_3$. The pseudoscalar can be used to construct *duals* of multivectors. The dual of a multivector $\boldsymbol{A}$ is

$$\boldsymbol{A}^* = \boldsymbol{A}\boldsymbol{I}_E^{-1} = \boldsymbol{A} \cdot \boldsymbol{I}_E^{-1} \tag{4.35}$$

For the Euclidean space $\mathbb{R}^3$ the inverse of the pseudoscalar is defined as

$$\begin{aligned} \boldsymbol{I}_E^{-1} &= \frac{\widetilde{\boldsymbol{I}}_E}{|\boldsymbol{I}_E|^2} = \widetilde{\boldsymbol{I}}_E \\ &= \boldsymbol{e}_3 \wedge \boldsymbol{e}_2 \wedge \boldsymbol{e}_1 = -\boldsymbol{I}_E \end{aligned} \tag{4.36}$$

Note that also the dual can be reversed

$$\begin{aligned} \boldsymbol{A}^* \cdot \boldsymbol{I}_E = \boldsymbol{A}^* \boldsymbol{I}_E &= \boldsymbol{A}\boldsymbol{I}_E^{-1}\boldsymbol{I}_E \\ &= \boldsymbol{A} \end{aligned} \tag{4.37}$$

The dual of a dual can also be calculated as

$$\begin{aligned} (\boldsymbol{A}^*)^* &= (\boldsymbol{A}\boldsymbol{I}_E^{-1})\boldsymbol{I}_E^{-1} \\ &= -(-\boldsymbol{A}\boldsymbol{I}_E)\boldsymbol{I}_E \\ &= \boldsymbol{A}\boldsymbol{I}_E^2 = -\boldsymbol{A} \end{aligned} \tag{4.38}$$

The dual of the different base elements are

$$\begin{aligned} 1^* &= -\boldsymbol{e}_{123} & \boldsymbol{e}_{123}^* &= 1 \\ \boldsymbol{e}_1^* &= -\boldsymbol{e}_{23} & \boldsymbol{e}_{23}^* &= \boldsymbol{e}_1 \\ \boldsymbol{e}_2^* &= -\boldsymbol{e}_{31} & \boldsymbol{e}_{31}^* &= \boldsymbol{e}_2 \\ \boldsymbol{e}_3^* &= -\boldsymbol{e}_{12} & \boldsymbol{e}_{12}^* &= \boldsymbol{e}_3 \end{aligned} \tag{4.39}$$

4.2 Conformal Geometric Space

The conformal geometric space is constructed by adding two extra dimension, $\boldsymbol{e}_+$ and $\boldsymbol{e}_-$, which are orthonormal to each other and the vectors in the Euclidean space, $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$, which is written

$$\boldsymbol{e}_+ \cdot \boldsymbol{e}_i = 0 \tag{4.40}$$

$$\boldsymbol{e}_- \cdot \boldsymbol{e}_i = 0 \tag{4.41}$$

$$\boldsymbol{e}_+ \cdot \boldsymbol{e}_- = 0 \tag{4.42}$$

for $i = 1, 2, 3$. It is given that

$$\boldsymbol{e}_+^2 = 1 \qquad \boldsymbol{e}_-^2 = -1 \tag{4.43}$$

Since the conformal geometric space is defined this way, it is written as $\mathbb{R}^{4,1}$.

These two basis vectors are usually replaced by two other orthonormal basis vectors, $\boldsymbol{e}_0$ and $\boldsymbol{e}_\infty$, defined by

$$\boldsymbol{e}_0 = \frac{1}{2}(\boldsymbol{e}_- - \boldsymbol{e}_+), \qquad \boldsymbol{e}_\infty = \boldsymbol{e}_- + \boldsymbol{e}_+ \tag{4.44}$$

Here $\boldsymbol{e}_0$ represents an arbitrary point at the origin, and $\boldsymbol{e}_\infty$ represents the point at infinity. It follows from the properties of $\boldsymbol{e}_+$ and $\boldsymbol{e}_-$ that

$$\boldsymbol{e}_0^2 = \boldsymbol{e}_\infty^2 = 0, \qquad \boldsymbol{e}_0 \cdot \boldsymbol{e}_\infty = -1 \tag{4.45}$$

The basis vectors of the conformal geometric space of a Euclidean space $\mathbb{R}^3$ is then given as

$$\{\boldsymbol{e}_0, \boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_\infty\} \tag{4.46}$$

The basis of the conformal geometric space is

$$\begin{aligned}
&\{1,\\
&\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \boldsymbol{e}_0, \boldsymbol{e}_\infty,\\
&\boldsymbol{e}_{23}, \boldsymbol{e}_{31}, \boldsymbol{e}_{12}, \boldsymbol{e}_{10}, \boldsymbol{e}_{20}, \boldsymbol{e}_{30}, \boldsymbol{e}_{1\infty}, \boldsymbol{e}_{2\infty}, \boldsymbol{e}_{3\infty}, \boldsymbol{e}_{0\infty},\\
&\boldsymbol{e}_{123}, \boldsymbol{e}_{120}, \boldsymbol{e}_{230}, \boldsymbol{e}_{310}, \boldsymbol{e}_{12\infty}, \boldsymbol{e}_{23\infty}, \boldsymbol{e}_{31\infty}, \boldsymbol{e}_{10\infty}, \boldsymbol{e}_{20\infty}, \boldsymbol{e}_{30\infty},\\
&\boldsymbol{e}_{1230}, \boldsymbol{e}_{123\infty}, \boldsymbol{e}_{230\infty}, \boldsymbol{e}_{310\infty}, \boldsymbol{e}_{120\infty},\\
&\boldsymbol{e}_{1230\infty}\}
\end{aligned} \tag{4.47}$$

which is a total of 32 elements.

4.2.1 Duals and Pseudoscalar

The pseudoscalar in conformal geometric algebra is

$$\boldsymbol{I}_C = \boldsymbol{e}_0 \wedge \boldsymbol{e}_1 \wedge \boldsymbol{e}_2 \wedge \boldsymbol{e}_3 \wedge \boldsymbol{e}_\infty \tag{4.48}$$

And the inverse of the pseudoscalar has the same property as that of $\boldsymbol{I}_E^{-1}$

$$\begin{aligned}
\boldsymbol{I}_C^{-1} &= \frac{\widetilde{\boldsymbol{I}}_C}{|\boldsymbol{I}_C|^2} = \widetilde{\boldsymbol{I}}_C \\
&= \boldsymbol{e}_\infty \wedge \boldsymbol{e}_3 \wedge \boldsymbol{e}_2 \wedge \boldsymbol{e}_1 \wedge \boldsymbol{e}_0 = -\boldsymbol{I}_C
\end{aligned} \tag{4.49}$$

The dual in conformal geometric algebra is defined as

$$\boldsymbol{A}^* = \boldsymbol{A}\boldsymbol{I}_C^{-1} \tag{4.50}$$

where the dual can be reversed

$$\begin{aligned}
\boldsymbol{A}^* \cdot \boldsymbol{I}_C = \boldsymbol{A}^* \boldsymbol{I}_C &= \boldsymbol{A}\boldsymbol{I}_C^{-1}\boldsymbol{I}_C \\
&= \boldsymbol{A}
\end{aligned} \tag{4.51}$$

4.2.2 Representation of Euclidean Objects

In conformal geometric algebra there is a set of geometric objects that can be constructed using the inner product and outer product. These objects are characterize as either flats or rounds [19], where flats are flat objects, such as lines and planes, and round objects are round, such as circles and spheres.

Points

A Euclidean point $\boldsymbol{p} \in \mathbb{R}^3$ is defined as

$$\boldsymbol{p} = p_1\boldsymbol{e}_1 + p_2\boldsymbol{e}_2 + p_3\boldsymbol{e}_3 \tag{4.52}$$

and has the column vector representation

$$[\boldsymbol{p}] = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} \tag{4.53}$$

In conformal geometric algebra, the conformal geometric point $\boldsymbol{P}$ represents the point $\boldsymbol{p}$ with the function

$$\boldsymbol{P} = \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^2\boldsymbol{e}_\infty + \boldsymbol{e}_0 \tag{4.54}$$

and has the column vector representation

$$[\boldsymbol{P}] = \begin{bmatrix} [\boldsymbol{p}] \\ \frac{1}{2}\boldsymbol{p}^2 \\ 1 \end{bmatrix} \tag{4.55}$$

It is noted that the scaling of a conformal point $\alpha\boldsymbol{P} = \alpha\boldsymbol{p} + \frac{1}{2}\alpha\boldsymbol{p}^2\boldsymbol{e}_\infty + \alpha\boldsymbol{e}_0$ will represent the same Euclidean point $\boldsymbol{p}$. This means that $\boldsymbol{P}$ and $\alpha\boldsymbol{P}$ are equivalent for all $\alpha \neq 0$.

A property of the conformal point $\boldsymbol{P}$ is that if the point $\boldsymbol{p}$ is at the origin, $\boldsymbol{p} = 0$, then $\boldsymbol{P} = \boldsymbol{e}_0$ results in the point at the origin. If $\boldsymbol{p}$ is infinitely far away from the origin, then $\boldsymbol{P} = \boldsymbol{e}_\infty$.

The Euclidean distance between two points $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$ can be found by using the inner product between $\boldsymbol{P}_A$ and $\boldsymbol{P}_B$, which is seen from

$$\begin{aligned}
\boldsymbol{P}_A \cdot \boldsymbol{P}_B &= (\boldsymbol{p}_A + \frac{1}{2}\boldsymbol{p}_A^2\boldsymbol{e}_\infty + \boldsymbol{e}_0) \cdot (\boldsymbol{p}_B + \frac{1}{2}\boldsymbol{p}_B^2\boldsymbol{e}_\infty + \boldsymbol{e}_0) \\
&= \boldsymbol{p}_A \cdot \boldsymbol{p}_B + \frac{1}{2}\boldsymbol{p}^2\boldsymbol{p}_A \cdot \boldsymbol{e}_\infty + \boldsymbol{p}_A \cdot \boldsymbol{e}_0 + \frac{1}{2}\boldsymbol{p}_A^2\boldsymbol{e}_\infty \cdot \boldsymbol{p}_B \\
&+ \frac{1}{4}\boldsymbol{p}_A^2\boldsymbol{p}_B^2\boldsymbol{e}_\infty \cdot \boldsymbol{e}_\infty + \frac{1}{2}\boldsymbol{p}_A^2\boldsymbol{e}_\infty \cdot \boldsymbol{e}_0 + \boldsymbol{e}_0 \cdot \boldsymbol{p}_B + \frac{1}{2}\boldsymbol{p}_B^2\boldsymbol{e}_0 \cdot \boldsymbol{e}_\infty + \boldsymbol{e}_0 \cdot \boldsymbol{e}_0 \\
&= \boldsymbol{p}_A \cdot \boldsymbol{p}_B - \frac{1}{2}\boldsymbol{p}_A^2 - \frac{1}{2}\boldsymbol{p}_B^2 \\
&= -\frac{1}{2}(\boldsymbol{p}_A^2 - 2\boldsymbol{p}_A \cdot \boldsymbol{p}_B + \boldsymbol{p}_B^2) \\
&= -\frac{1}{2}(\boldsymbol{p}_A - \boldsymbol{p}_B)^2
\end{aligned} \tag{4.56}$$

Rounds

The two-dimensional round is the circle, and is given by the outer product between three points

$$\boldsymbol{C} = \boldsymbol{P}_A \wedge \boldsymbol{P}_B \wedge \boldsymbol{P}_C \tag{4.57}$$

The sphere is the three-dimensional round, and is the outer product between four points

$$\boldsymbol{S} = \boldsymbol{P}_A \wedge \boldsymbol{P}_B \wedge \boldsymbol{P}_C \wedge \boldsymbol{P}_D \tag{4.58}$$

The dual of a sphere can be written as

$$\boldsymbol{S}^* = \boldsymbol{P} - \frac{1}{2}r^2\boldsymbol{e}_\infty \tag{4.59}$$

where $\boldsymbol{P}$ is the center of the sphere and r is the radius. The column vector for $\boldsymbol{S}^*$ is written as

$$[\boldsymbol{S}^*] = \begin{bmatrix} \boldsymbol{p} \\ \frac{1}{2}(\boldsymbol{p}^2 - r^2) \\ 1 \end{bmatrix} \tag{4.60}$$

It is worth noting that if $r = 0$, $\boldsymbol{S}^*$ is identical to $\boldsymbol{P}$, which means that a point is a sphere with zero radius.

The radius of a sphere can be found with the dual of the sphere squared

$$\begin{aligned} \boldsymbol{S}^{*2} = \boldsymbol{S}^* \cdot \boldsymbol{S}^* &= (\boldsymbol{P} - \frac{1}{2}r^2\boldsymbol{e}_\infty) \cdot (\boldsymbol{P} - \frac{1}{2}r^2\boldsymbol{e}_\infty) \\ &= \boldsymbol{P} \cdot (\boldsymbol{P} - \frac{1}{2}r^2\boldsymbol{e}_\infty) - \frac{1}{2}r^2\boldsymbol{e}_\infty \cdot (\boldsymbol{P} - \frac{1}{2}r^2\boldsymbol{e}_\infty) \\ &= \boldsymbol{P} \cdot \boldsymbol{P} - \boldsymbol{P} \cdot \frac{1}{2}r^2\boldsymbol{e}_\infty - \boldsymbol{P} \cdot \frac{1}{2}r^2\boldsymbol{e}_\infty + \frac{1}{2}r^2\boldsymbol{e}_\infty \cdot \frac{1}{2}r^2\boldsymbol{e}_\infty \\ &= 0 + \frac{1}{2}r^2 + \frac{1}{2}r^2 + 0 \\ &= r^2 \end{aligned} \tag{4.61}$$

The one-dimensional round is known as a point pair, is constructed by taking the outer product between two points

$$\boldsymbol{Q} = \boldsymbol{P}_A \wedge \boldsymbol{P}_B \tag{4.62}$$

Each point in the point pair can be found by using

$$\boldsymbol{P}_\pm = \frac{\boldsymbol{Q} \pm \sqrt{\boldsymbol{Q}^2}}{-\boldsymbol{e}_\infty \cdot \boldsymbol{Q}} \tag{4.63}$$

An important observation is that the dual of a circle is a point pair and vice versa, and they have the same radius. This can be thought of with having a sphere, where the two "poles" of the sphere generate the point pair. If so, then the "equator" of the sphere is the dual of the point pair, and is indeed a circle.

Flats

Flats are all Euclidean objects that are flat, or have no curvature.

A plane is a three-dimensional flat, and is the outer product between three points and the point at infinity

$$\boldsymbol{\Pi} = \boldsymbol{P}_A \wedge \boldsymbol{P}_B \wedge \boldsymbol{P}_C \wedge \boldsymbol{e}_\infty \tag{4.64}$$

The dual of a plane can be written as

$$\boldsymbol{\Pi}^* = \boldsymbol{n} + \delta\boldsymbol{e}_\infty \tag{4.65}$$

where $\boldsymbol{n}$ is the normal of the plane and δ is the distance between the plane and the origin. The column vector for $\boldsymbol{\Pi}^*$ is written as

$$[\boldsymbol{\Pi}^*] = \begin{bmatrix} \boldsymbol{n} \\ \delta \\ 0 \end{bmatrix} \tag{4.66}$$

The distance between a point $\boldsymbol{P}$ and a plane $\boldsymbol{\Pi}$ is found from

$$\begin{aligned} \boldsymbol{P} \cdot \boldsymbol{\Pi}^* &= (\boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^2\boldsymbol{e}_\infty + \boldsymbol{e}_0) \cdot (\boldsymbol{n} + \delta\boldsymbol{e}_\infty) \\ &= \boldsymbol{p} \cdot \boldsymbol{n} + \delta\boldsymbol{p} \cdot \boldsymbol{e}_\infty + \frac{1}{2}\boldsymbol{p}^2\boldsymbol{e}_\infty \cdot \boldsymbol{n} \\ &+ \frac{1}{2}\boldsymbol{p}^2\delta\boldsymbol{e}_\infty \cdot \boldsymbol{e}_\infty + \boldsymbol{e}_0 \cdot \boldsymbol{n} + \delta\boldsymbol{e}_0 \cdot \boldsymbol{e}_\infty \\ &= \boldsymbol{p} \cdot \boldsymbol{n} - \delta \end{aligned} \tag{4.67}$$

The two-dimensional flat is a line, and is the outer product between two point and the point at infinity

$$\boldsymbol{L} = \boldsymbol{P}_A \wedge \boldsymbol{P}_B \wedge \boldsymbol{e}_\infty \tag{4.68}$$

The one-dimensional flat is known as a flat point, and is the outer product between a point and the point at infinity

$$\boldsymbol{\phi} = \boldsymbol{P}_A \wedge \boldsymbol{e}_\infty \tag{4.69}$$

4.2.3 Intersections

In conformal geometric algebra, it is possible to calculate the intersection between various geometric objects. The intersection between the two geometric objects $\boldsymbol{A}$ and $\boldsymbol{B}$ is defined as

$$\boldsymbol{M}^* = \boldsymbol{A}^* \wedge \boldsymbol{B}^* \tag{4.70}$$

where $\boldsymbol{M}$ is referred to as the meet. This also means that

$$\begin{aligned} \boldsymbol{M} = \boldsymbol{M}^* \cdot \boldsymbol{I}_C &= (\boldsymbol{A}^* \wedge \boldsymbol{B}^*) \cdot \boldsymbol{I}_C \\ &= (\boldsymbol{A}\boldsymbol{I}_C^{-1} \wedge \boldsymbol{B}\boldsymbol{I}_C^{-1}) \cdot \boldsymbol{I}_C \\ &= \boldsymbol{A}\boldsymbol{I}_C^{-1} \cdot (\boldsymbol{B}\boldsymbol{I}_C^{-1} \cdot \boldsymbol{I}_C) \\ &= \boldsymbol{A}^* \cdot \boldsymbol{B} \end{aligned} \tag{4.71}$$

Which comes from a property of blades that is

$$(\boldsymbol{A}_{\langle k \rangle} \wedge \boldsymbol{B}_{\langle l \rangle}) \cdot \boldsymbol{C}_{\langle m \rangle} = \boldsymbol{A}_{\langle k \rangle} \cdot (\boldsymbol{B}_{\langle l \rangle} \cdot \boldsymbol{C}_{\langle m \rangle}) \tag{4.72}$$

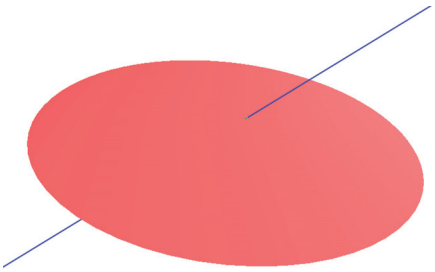
where $\boldsymbol{A}_{\langle k \rangle}$ is a k-blade, $\boldsymbol{B}_{\langle l \rangle}$ is a l blade, $\boldsymbol{C}_{\langle m \rangle}$ is a m-blade, and that $1 \leq k, l, m \leq n$ and $m \geq k + l$.

An example of this is the intersection between a line $\boldsymbol{L}$ and a plane $\boldsymbol{\Pi}$. The intersection between the two is

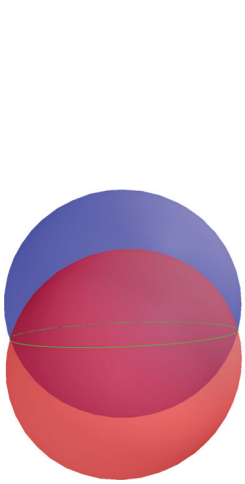
$$\begin{aligned}
\boldsymbol{M} &= \boldsymbol{\Pi}^* \cdot \boldsymbol{L} \\
&= \boldsymbol{\Pi}^* \cdot (\boldsymbol{P}_A \wedge \boldsymbol{P}_B \wedge \boldsymbol{e}_\infty) \\
&= (\boldsymbol{\Pi}^* \cdot \boldsymbol{P}_A)(\boldsymbol{P}_B \wedge \boldsymbol{e}_\infty) - (\boldsymbol{\Pi}^* \cdot \boldsymbol{P}_B)(\boldsymbol{P}_A \wedge \boldsymbol{e}_\infty) + (\boldsymbol{\Pi}^* \cdot \boldsymbol{e}_\infty)(\boldsymbol{P}_A \wedge \boldsymbol{P}_B) \\
&= ((\boldsymbol{n} + \delta \boldsymbol{e}_\infty) \cdot \boldsymbol{P}_A)(\boldsymbol{P}_B \wedge \boldsymbol{e}_\infty) - ((\boldsymbol{n} + \delta \boldsymbol{e}_\infty) \cdot \boldsymbol{P}_B)(\boldsymbol{P}_A \wedge \boldsymbol{e}_\infty) \\
&= d_A(\boldsymbol{P}_B \wedge \boldsymbol{e}_\infty) - d_B(\boldsymbol{P}_A \wedge \boldsymbol{e}_\infty) \\
&= (d_A \boldsymbol{P}_B - d_B \boldsymbol{P}_A) \wedge \boldsymbol{e}_\infty
\end{aligned} \tag{4.73}$$

where d_A and d_B are the distances between the plane $\boldsymbol{\Pi}$ and the points $\boldsymbol{p}_A$ and $\boldsymbol{p}_B$ respectively. Note that the resulting calculation is a flat point, which with some further calculations can be defined as $\boldsymbol{M} = \boldsymbol{P}_C \wedge \boldsymbol{e}_\infty$, where $\boldsymbol{P}_C$ is the intersecting point between $\boldsymbol{\Pi}$ and $\boldsymbol{L}$. An example is shown in Figure 4.2.

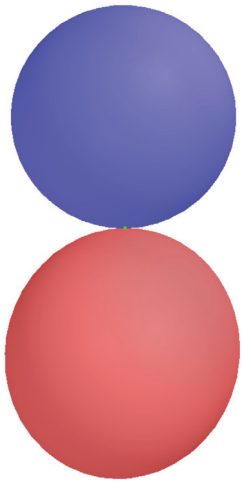
Another example is the intersection between two spheres, where if they intersect, the resulting intersection is a circle. If the spheres only touch at the surface, the resulting intersection is a point, and if they do not intersect, the result is an imaginary circle. This is shown in Figure 4.2.



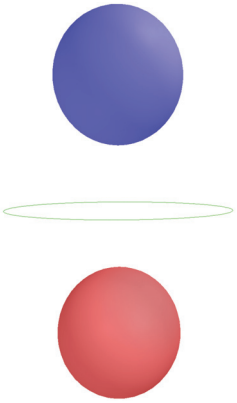
(a) Intersection between a line and a plane. The intersecting float point is seen in green.



(b) The intersection between two spheres shown as a green circle.



(c) The intersection between two spheres shown as a green point.



(d) The intersection between two spheres shown as a green imaginary circle.

Figure 4.2: Examples of different intersections.

Chapter 5

Applications of Conformal Geometric Algebra

How did the tree feel in spring?

Releaved.

5.1 Sphere Fitting

This section explains the theory behind the sphere fitting method presented in [18], which finds an op timal sphere estimate based on a set of points. The section starts by explaining the Sampson distance [52], which is a least-squares function which is minimized when fitting spheres to points, followed by the Rayleigh quotient, which explains how the Sampson distance relates to the generalized eigenvalue problem, and the optimal solution to the eigenvalue problem. The Pratt fit is then presented, which is a circle fitting algorithm using the method described above, followed by how this can be extended to become a sphere fitting algorithm using conformal geometric algebra.

5.1.1 Sampson Distance

Consider the circle $g(\boldsymbol{x}) = 0$ in the xy-plane, where

$$g(\boldsymbol{x}) = a(x^2 + y^2) + dx + ey + f = 0 \tag{5.1}$$

and $\boldsymbol{x} = [x, y]^{\mathrm{T}}$. Note that $a \neq 0$ for circles, and $a = 0$ for lines. The gradient is

$$\nabla g(\boldsymbol{x}) = \begin{bmatrix} 2ax + d \\ 2ay + e \\ 0 \end{bmatrix} = 2a \begin{bmatrix} x - x_0 \\ y - y_0 \\ 0 \end{bmatrix} \tag{5.2}$$

where $x_0 = -\frac{d}{2a}$ and $y_0 = -\frac{e}{2a}$ are the coordinates of the center of the circle. The gradient $\nabla g(\boldsymbol{x})$ at the point $\boldsymbol{x}$, which is on the circle, has the direction from the

center of the circle, x_0, to the point

$$\boldsymbol{x} = \boldsymbol{x}_0 + \frac{\nabla g(\boldsymbol{x})}{2a} \tag{5.3}$$

where $\boldsymbol{x_0} = [x_0, y_0]^{\mathrm{T}}$. The norm of the gradient is

$$\begin{aligned} |\nabla g(\boldsymbol{x})|^2 &= (2ax + d)^2 + (2ay + e)^2 \\ &= 4a^2x^2 + 4adx + d^2 + 4a^2y^2 + 4aey + e^2 \\ &= 4a(a(x^2 + y^2) + dx + ey + f) + d^2 + e^2 - 4af \end{aligned} \tag{5.4}$$

Consider the point $\boldsymbol{x}_i$, which is not on the circle $g(\boldsymbol{x})$. The distance from $\boldsymbol{x}_i$ to the $\boldsymbol{x}$ can be approximated by

$$g(\boldsymbol{x}_i) = g(\boldsymbol{x}) + \nabla g(\boldsymbol{x})(\boldsymbol{x}_i - \boldsymbol{x}) \tag{5.5}$$

Since $g(\boldsymbol{x}) = 0$, it can be rewritten as

$$g(\boldsymbol{x}_i) = \nabla g(\boldsymbol{x})(\boldsymbol{x}_i - \boldsymbol{x}) \tag{5.6}$$

and it follows that

$$|\boldsymbol{x}_i - \boldsymbol{x}|^2 = \frac{g(\boldsymbol{x}_i)^2}{|\nabla g(\boldsymbol{x})|^2} = \delta_i \tag{5.7}$$

The distance δ_i is called the Sampson distance [52].

From (5.1) it is seen that $a(x^2 + y^2) + dx + ey + f = 0$ so the magnitude of the gradient will satisfy

$$|\nabla g(\boldsymbol{x})|^2 = d^2 + e^2 + 4af = 4a^2r^2 \tag{5.8}$$

where r is the radius of the circle. The Sampson distance of the circle to the point $\boldsymbol{x}_i$ will then be

$$\delta_i^2 = \frac{g(\boldsymbol{x}_i)^2}{|\nabla g(\boldsymbol{x})|^2} = \frac{(a(x_i^2 + y_i^2) + dx_i + ey_i + f)^2}{d^2 + e^2 - 4af} \tag{5.9}$$

This expression of the Sampson distance is used in the Pratt fit [41].

5.1.2 The Rayleigh quotient

Consider the minimization of the function

$$f(\boldsymbol{x}) = \frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}}{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}} \tag{5.10}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{P}$ and $\boldsymbol{Q}$ are positive definite symmetric $n \times n$ matrices. It is assumed that $\boldsymbol{P}$ is positive definite, which means that $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x} > 0$ for all $\boldsymbol{x} \neq 0$.

The condition for optimality is

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = 2\frac{\boldsymbol{P}\boldsymbol{x}(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}) - \boldsymbol{Q}\boldsymbol{x}(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x})}{(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x})^2} = 0 \tag{5.11}$$

This gives

$$\boldsymbol{P}\boldsymbol{x}(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}) = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x})\boldsymbol{Q}\boldsymbol{x} \tag{5.12}$$

and it follows that the minimum is achieved when

$$\boldsymbol{P}\boldsymbol{x} = f(\boldsymbol{x})\boldsymbol{Q}\boldsymbol{x} \tag{5.13}$$

The solution for this problem is found from the generalized eigenvalue problem

$$\boldsymbol{P}\boldsymbol{x} = \lambda\boldsymbol{Q}\boldsymbol{x} \tag{5.14}$$

The generalized eigenvalues λ_i and the corresponding eigenvectors $\mathbf{v}_i$ satisfies $(\boldsymbol{P} - \lambda_i\boldsymbol{Q})\mathbf{v}_i = 0$ for $i = 1, \ldots, n$. The optimal solution is then $\boldsymbol{x}^* = k\mathbf{v}_i^*$ where $k \neq 0$ is a scaling factor the can be selected freely, and $\mathbf{v}_i^*$ is the eigenvector which corresponds to the smallest positive eigenvalue λ_i^*. Moreover, the optimal solution is $f(\boldsymbol{x}^*) = \lambda_i^*$.

5.1.3 The Generalized Eigenvalue Problem

The generalized eigenvalue problem is written

$$\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{B}\boldsymbol{x} \tag{5.15}$$

where $\boldsymbol{A} = \boldsymbol{A}^{\mathrm{T}}$ and $\boldsymbol{B} = \boldsymbol{B}^{\mathrm{T}}$ are symmetric $n \times n$ matrices, and $\boldsymbol{B}$ is positive definite. Then the matrix $\boldsymbol{B}$ has n positive eigenvalues $\mu_i > 0$ for $i = 1, \ldots, n$ and n orthonormal eigenvectors $\boldsymbol{m}_i$ so that $\boldsymbol{B}\boldsymbol{m}_i = \mu\boldsymbol{m}_i$. The matrix $\boldsymbol{M} = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n]$ is orthogonal, and satisfies $\boldsymbol{M}^{\mathrm{T}}\boldsymbol{M} = \boldsymbol{I}$. It follows that $\boldsymbol{B}\boldsymbol{M} = \boldsymbol{D}\boldsymbol{M}$, where $\boldsymbol{D} = \mathrm{diag}(\mu_1, \ldots, \mu_n)$, and $\boldsymbol{B}$ can be transformed to a diagonal form by

$$\boldsymbol{M}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{M} = \boldsymbol{D} = \mathrm{diag}(\mu_1, \ldots, \mu_n) \tag{5.16}$$

The generalized eigenvalue problem can then be transformed into the form

$$\boldsymbol{M}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{M}\boldsymbol{y} = \lambda\boldsymbol{D}\boldsymbol{y} \tag{5.17}$$

where $\boldsymbol{x} = \boldsymbol{M}\boldsymbol{y}$. Define $\boldsymbol{N} = \boldsymbol{D}^{-1/2}\boldsymbol{M}$ where $\boldsymbol{D}^{-1/2} = \mathrm{diag}(\mu_1^{-1/2}, \ldots, \mu_n^{-1/2})$, which gives $\boldsymbol{N}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{N} = \boldsymbol{I}$. Then the generalized eigenvalue problem becomes

$$\boldsymbol{N}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{N}\boldsymbol{z} = \lambda\boldsymbol{z} \tag{5.18}$$

where $\boldsymbol{x} = \boldsymbol{M}\boldsymbol{y} = \boldsymbol{N}\boldsymbol{z}$. It is seen that the eigenvalues of the generalized eigenvalue problem will be the same as the eigenvalues of $\boldsymbol{N}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{N}$.

From Sylvester's law of inertia it follows that the eigenvalues λ_i of $\boldsymbol{N}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{N}$ will be related to the eigenvalues γ_i of $\boldsymbol{A}$ so that there will be the same number of positive eigenvalues, negative eigenvalues and eigenvalues equals to zero.

If instead $\boldsymbol{A}$ is positive definite and $\boldsymbol{B}$ is positive semidefinite, then $\boldsymbol{B}$ will have at least one eigenvalue that is equal to zero. This implies that the matrix $\boldsymbol{N}$ becomes undefined. In this case the problem can be reformulated as

$$\boldsymbol{B}\boldsymbol{x} = \frac{1}{\lambda}\boldsymbol{A}\boldsymbol{x} \tag{5.19}$$

Then it follows that the eigenvalues $\frac{1}{\lambda}$ will be related to the eigenvalues μ_i of $\boldsymbol{B}$ so that there will be the same number of positive and negative eigenvalues $1/\lambda_i$ as there are positive and negative μ_i respectively. If $\mu_n = 0$, then the corresponding eigenvalue λ_n will be undefined.

If the case where $\mu_i > 0$ for $i = 1, \ldots, n-1$, and $\mu_n = 0$, then it follows that there will be $n-1$ positive eigenvalues λ_i for $i = 1, \ldots, n-1$ and one undefined eigenvalue λ_n. It follows that the smallest eigenvalue is $\lambda_{n-1} > 0$.

5.1.4 Eigenvalues and the Rayleigh quotient

Consider the minimization of the function

$$f(\boldsymbol{x}) = \frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}}{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x}} \tag{5.20}$$

where $\boldsymbol{x} \in \mathbb{R}^n$. Suppose that $\boldsymbol{P} = \boldsymbol{P}^{\mathrm{T}} > 0$ is positive definite.

A necessary condition for the minimum of $f(\boldsymbol{x})$ is that the optimal solution $\boldsymbol{x}^*$ is an eigenvector $\mathbf{v}_i^*$ of the generalized eigenvalue problem $\boldsymbol{P}\boldsymbol{x} = \lambda\boldsymbol{Q}\boldsymbol{x}$. The minimum value of the function is then $f(\boldsymbol{x}^* = \lambda_i^*)$.

If $\boldsymbol{Q}$ has one negative eigenvalue and $n-1$ positive eigenvalues, then the generalized eigenvalue problem will have one negative eigenvalue and $n-1$ positive eigenvalues. The function $f(\boldsymbol{x})$ cannot be negative. This means that the minimum value of $f(\boldsymbol{x})$ is achieved with the smallest non-negative eigenvalue of the generalized eigenvalue problem.

If $\boldsymbol{Q}$ has one eigenvalue equal to zero and $n-1$ positive eigenvalues, then the generalized eigenvalue problem $\boldsymbol{Q}\boldsymbol{x} = \frac{1}{\lambda}\boldsymbol{P}\boldsymbol{x}$ will have one eigenvalue equal to zero, which means that the corresponding eigenvalue λ_i will be undefined. The minimum value of $f(\boldsymbol{x})$ is achieved with the smallest eigenvalue of the generalized eigenvalue problem.

5.1.5 The Pratt Fit

The Pratt fit [41, 14] is a circle fitting method based on the minimization of the Sampson error $\frac{g(\boldsymbol{x}}{|\nabla g(\boldsymbol{x})|}$ using the approximation (5.8) for the gradient. The objective function to be minimized is then

$$L_P(\boldsymbol{c}) = \sum_{i=1}^{n} \frac{g(\boldsymbol{x}_i)^2}{|\nabla g(\boldsymbol{x})|^2} = \sum_{i=1}^{n} \frac{(a(x_i^2 + y_i^2) + dx_i + ey_i + f)^2}{d^2 + e^2 - 4af} \tag{5.21}$$

where $\boldsymbol{c} = [a, d, e, f]^{\mathrm{T}}$. The objective function can be written as

$$L_P(\boldsymbol{c}) = \frac{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z}\boldsymbol{c}}{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{N}\boldsymbol{c}} = \frac{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{c}}{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{N}\boldsymbol{c}} \tag{5.22}$$

where $\boldsymbol{M} = \boldsymbol{Z}^{\mathrm{T}}\boldsymbol{Z}$ and

$$\boldsymbol{Z} = \begin{bmatrix} (x_1^2 + y_1^2) & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ (x_n^2 + y_n^2) & x_n & y_n & 1 \end{bmatrix} \tag{5.23}$$

and

$$\boldsymbol{N} = \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & 0 & 0 & 0 \end{bmatrix} \tag{5.24}$$

Here, the matrix $\boldsymbol{M}$ is positive definite, while $\boldsymbol{N}$ has eigenvalues $\lambda_{N1} = 2$, $\lambda_{N2} = \lambda_{N3} = 1$ and $\lambda_{N4} = -2$.

Based on the results from the Rayleigh quotient, the minimum is found from the generalized eigenvalue problem

$$\boldsymbol{M}\boldsymbol{c} = \lambda \boldsymbol{N}\boldsymbol{c} \tag{5.25}$$

Due to the eigenvalues of $\boldsymbol{N}$, this generalized eigenvalue problem has four eigenvalues $\lambda_1, \ldots, \lambda_4$, where $\lambda_1 \geq \lambda_2 \geq \lambda_3 > 0$ and $\lambda_4 < 0$.

The minimum of the objective function us therefore $L_P(\boldsymbol{c}^*) = \lambda_3$, which is found for the optimal solution $\boldsymbol{c}^* = \mathbf{v}_3$, where $\mathbf{v}_3$ corresponds to the smallest positive eigenvalue λ_3.

5.1.6 Sphere Fitting

In conformal geometric algebra, a sphere $\boldsymbol{S}$ can be fitted to a point cloud $\boldsymbol{p}_i$, $i = 1, \ldots, n$ as proposed in [18] by minimizing the objective function

$$f(\boldsymbol{S}^*) = \sum_{i=1}^{n} \frac{(\boldsymbol{S}^* \cdot \boldsymbol{P}_i)^2}{\boldsymbol{S}^{*2}} \tag{5.26}$$

where

$$\begin{aligned} \frac{(\boldsymbol{S}^* \cdot \boldsymbol{P}_i)^2}{\boldsymbol{S}^{*2}} &= \frac{1}{4}\frac{(r^2 - |\boldsymbol{p}_i - \boldsymbol{p}_{\boldsymbol{S}}|^2)^2}{r^2} \\ &= \frac{1}{4}\frac{(r^2 - 2r\delta_i - \delta_i^2 - r^2)^2}{r^2} \\ &= \frac{1}{4}\frac{(-\delta_i(2r - \delta_i))^2}{r^2} \\ &= \delta_i^2 \left(1 + \frac{\delta_i}{2r}\right)^2 \\ &= \delta_i^2 \left(1 + \frac{\delta_i}{r} + \left(\frac{\delta_i}{2r}\right)^2\right) \\ &\approx \delta^2 \end{aligned} \tag{5.27}$$

This method is an extension to the Pratt fit. The condition for optimality is

$$(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{P}_i(\boldsymbol{P}_i\cdot\boldsymbol{S}^*))\wedge\boldsymbol{S}^*=0 \tag{5.28}$$

This condition is satisfied when

$$\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{P}_i(\boldsymbol{P}_i\cdot\boldsymbol{S}^*)=\lambda\boldsymbol{S}^* \tag{5.29}$$

where $\lambda\neq 0$ is a scalar.

This follows as $\boldsymbol{a}\wedge\boldsymbol{b}=0\rightarrow\boldsymbol{a}=\gamma\boldsymbol{b}$. When $\boldsymbol{a}$ and $\boldsymbol{b}$ are non-zero vectors and $\gamma\neq 0$is a scalar. In coordinate form this gives

$$[\boldsymbol{Q}][\boldsymbol{S}^*]=\lambda[\boldsymbol{S}^*] \tag{5.30}$$

where

$$[\boldsymbol{Q}]=\frac{1}{n}\sum_{i=1}^{n}[\boldsymbol{P}_i][\boldsymbol{P}_i]^{\mathrm{T}}[\boldsymbol{M}] \tag{5.31}$$

where

$$[\boldsymbol{P}_i]=\begin{bmatrix}[\boldsymbol{p}_i]\\ \frac{1}{2}\boldsymbol{p}_i^2\\ 1\end{bmatrix},\qquad [\boldsymbol{M}]=\begin{bmatrix}1&0&0&0&0\\0&1&0&0&0\\0&0&1&0&0\\0&0&0&0&-1\\0&0&0&-1&0\end{bmatrix} \tag{5.32}$$

It is seen that the condition for optimality is that λ is an eigenvalue of $\boldsymbol{Q}$.

The objective function is

$$L=\frac{1}{n}\sum_{i=1}^{n}\frac{(\boldsymbol{P}_i\boldsymbol{S}^*)^2}{\boldsymbol{S}^{*2}}=\frac{[\boldsymbol{x}]^{\mathrm{T}}[\boldsymbol{P}_i][\boldsymbol{P}_i]^{\mathrm{T}}[\boldsymbol{S}^*]}{[\boldsymbol{S}^*]^{\mathrm{T}}[\boldsymbol{S}^*]} \tag{5.33}$$

$$=\frac{[\boldsymbol{S}^*]^{\mathrm{T}}[\boldsymbol{Q}][\boldsymbol{S}^*]}{[\boldsymbol{S}^*]^{\mathrm{T}}[\boldsymbol{S}^*]}=\frac{[\boldsymbol{S}^*]^{\mathrm{T}}\lambda[\boldsymbol{S}^*]}{[\boldsymbol{S}^*]^{\mathrm{T}}[\boldsymbol{S}^*]}=\lambda \tag{5.34}$$

This means that the minimum value of the objective function is achieved for the smallest positive eigenvalue λ. The sphere is then given by $[\boldsymbol{S}^*]=\alpha\mathbf{v}$, where α is a scalar that is selected to scale the expression for the sphere, and $\mathbf{v}$ is the eigenvector corresponding to the eigenvalue λ.

5.2 RANSAC

RANdom SAmpling Consensus (RANSAC) [21] is an iterative model fitting method, which can be efficiently used even with a data set with many outlier. The method starts with a mathematical model, and finds which data points support the model and which do not. The mathematical model is generated from a random set of points.

For each iteration the steps of the RANSAC algorithm is as follows

1. Randomly sample a minimum subset of data points $\hat{\boldsymbol{x}}_k$ from the data set $\boldsymbol{X}$, that are used to generate the model candidate $\boldsymbol{M}_k(\hat{\boldsymbol{x}}_k)$. Here, k is the iteration of the RANSAC algorithm
2. Categorize all the data points either as inliers or outliers. Inliers support the model candidate $\boldsymbol{M}_k(\hat{\boldsymbol{x}_k})$, while outliers oppose it.
3. If $\boldsymbol{M}_k$ has more inliers than the currently optimal model candidate $\boldsymbol{M}_*$, then $\boldsymbol{M}_k$ becomes the new optimal solution, $\boldsymbol{M}_* = \boldsymbol{M}_k$.

There are four parameters that are required to develop a specific RANSAC algorithm

- $\boldsymbol{M}_k(\hat{\boldsymbol{x}_k})$, which is the function for generating the mathematical model
- n, which is the number of iterations.
- $f(\boldsymbol{M}_k, \boldsymbol{x_i})$, which finds the error between the model $\boldsymbol{M}_k$ and the point $\boldsymbol{x}_i \in \boldsymbol{X}$.
- E_T, which defines the error tolerance which is required for a point to be classified as an inlier, so that $f(\boldsymbol{M}_k, \boldsymbol{x}_i) < E_T$.

5.2.1 RANSAC and Conformal Geometric Algebra

For point cloud analysis, RANSAC is often used to define a geometric object that fits the points in the point cloud. An example of this is in [55], where the 3D camera takes pictures of a scene, and that scene contains a table with several objects on top. The interesting points are those of the objects, but most of the points represents the table. In order to perform a segmentation, a RANSAC method is set up so that it tries to fit the 3D point cloud to a plane, which will separate inliers, the points that represent the table; and the outliers, the points that represents the objects on top of the table.

In order to construct a model candidate of a plane in conformal geometric algebra, three points are required. This means that for each iteration, three random points are selected, $\hat{\boldsymbol{x}}_k = \{\boldsymbol{P}_{Ak}, \boldsymbol{P}_{Bk}, \boldsymbol{P}_{Ck}\}$, where $\boldsymbol{P}_{jk}$ is the conformal geometric point representing the Euclidean point $\boldsymbol{x}_j$, for $j = \{A, B, C\}$.

The plane model candidate $\boldsymbol{M}_k$ is constructed as

$$\boldsymbol{M}_k(\hat{\boldsymbol{x}}_k) = \boldsymbol{P}_{Ak} \wedge \boldsymbol{P}_{Bk} \wedge \boldsymbol{P}_{Ck} \wedge \boldsymbol{e}_\infty \tag{5.35}$$

and the error function $f(\boldsymbol{M}_k, \boldsymbol{x}_i)$ is

$$|\boldsymbol{M}_k^* \cdot \boldsymbol{P}_{\boldsymbol{x}_i}| < E_T \tag{5.36}$$

where $\boldsymbol{M}_k$ is a conformal geometric plane, and $\boldsymbol{P}_{\boldsymbol{x}_i}$ is the conformal geometric point of the Euclidean point $\boldsymbol{x}_i$.

5.3 Inverse Kinematics

Analytical inverse kinematics is a well-developed problem in robotics. Solutions are available as text-book material for revolute robots with a spherical wrist, or with three consecutive parallel axes [53, 54]. The solutions are given in terms of

trigonometric expressions, which are straightforward to find, although they can be somewhat involved. The complexity of the equations is partly related to the book-keeping of the different solutions related to shoulder left or right, elbow up or down, and wrist flipped or not.

Conformal geometric algebra provides additional insight into the problem. The inverse kinematics has been previously solved for a robot with 5 revolute joints in terms of spheres, planes and lines, and the intersection of these geometric objects [26, 27, 61]. This is extended in this thesis to an inverse kinematics solution for the Kuka KR6 R900 sixx and the Universal Robots UR5.

An example of the advantage granted by conformal geometric algebra is when determining elbow up and elbow down. Assume a robot arm with a known position of the wrist, $\boldsymbol{P}_w$, and the base, $\boldsymbol{P}_b$, and an upper arm with a length of d_u and a lower arm with a length of d_l, where the upper arm is attached to the base and the lower arm attached to the wrist, see Figure 5.1a.

The range of solutions for the elbow joint given the upper arm is the a circle with a radius d_u from the base, while for the lower arm it is a circle with a radius d_l from the wrist. On a standardized 6DOF robot, the elbow, wrist and base lie on the same plane $\boldsymbol{\Pi}$, which means that the two circles also lies on $\boldsymbol{\Pi}$.

Geometrically, this means that the solution to elbow position is where the two circles intersects, as seen in Figure 5.1b and Figure 5.1c. In conformal geometric algebra, the two circles can be generated by

$$\boldsymbol{C}_w = \boldsymbol{\Pi} \cdot (\boldsymbol{P}_w - \frac{1}{2}d_l^2) \qquad \boldsymbol{C}_b = \boldsymbol{\Pi} \cdot (\boldsymbol{P}_b - \frac{1}{2}d_u^2) \tag{5.37}$$

where $\boldsymbol{C}_w$ is the circle originating from the wrist point $\boldsymbol{P}_w$ with the radius d_l, while $\boldsymbol{C}_b$ is the circle originating from the base, $\boldsymbol{P}_b$, with the radius d_u. Note that the circles are generated by intersecting the plane $\boldsymbol{\Pi}$ with a sphere $\boldsymbol{S}^* = \boldsymbol{P} - \frac{1}{2}r^2$ using (4.71).

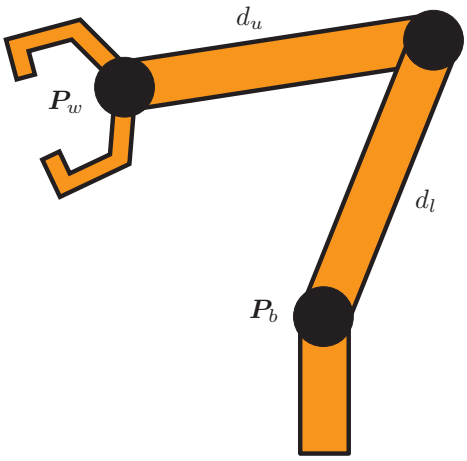
The intersection between the two circles is a point pair, which geometrically means that there are two points of intersections between the circles. These two intersections are the solutions to either elbow up or elbow down, and can in conformal geometric algebra be calculated as

$$\boldsymbol{Q} = \boldsymbol{C}_w^* \cdot \boldsymbol{C}_b \tag{5.38}$$

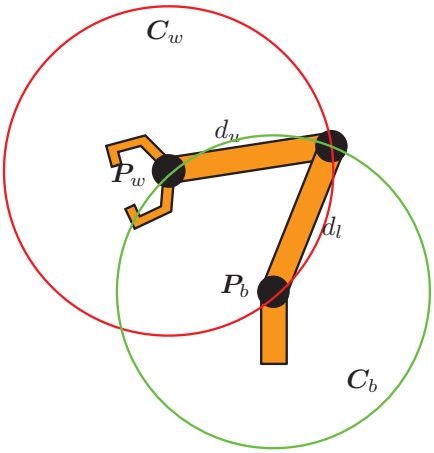
The point pair $\boldsymbol{Q}$ contains the full solution to the elbows position, and in that way makes it possible to chose which configuration of elbow up and elbow down that should be chosen. Note that the intersection between the two circles is calculated in the same way as (5.37), using (4.71).

To find the individual points in the point pair, we use (4.63) so that

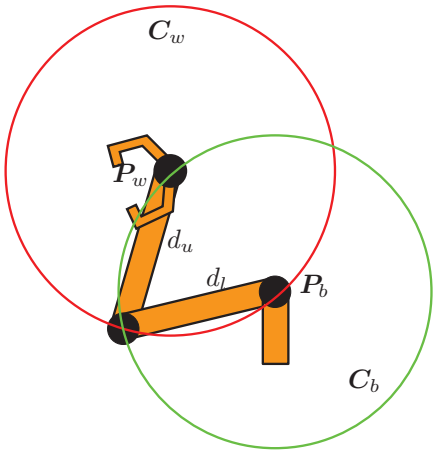
$$\boldsymbol{P}_\pm = \frac{\boldsymbol{Q} \pm \sqrt{\boldsymbol{Q}^2}}{-\boldsymbol{e}_\infty \cdot \boldsymbol{Q}} \tag{5.39}$$



(a) A general robot arm with an elbow up configuration.



(b) The robot arm with an elbow up configuration. The elbow joint is at the intersection of $\boldsymbol{C}_w$ and $\boldsymbol{C}_b$.



(c) The robot arm with an elbow down configuration. The elbow joint is at the intersection of $\boldsymbol{C}_w$ and $\boldsymbol{C}_b$.

Figure 5.1: The intersection of the two spheres $\boldsymbol{C}_w$ and $\boldsymbol{C}_b$ shows the solution to the elbow joint.

where $\boldsymbol{P}_{\pm}$ are the elbow up and elbow down configurations. With this we can define the configuration variable k_e so that

$$k_e = \begin{cases} 1, & \text{Elbow up} \\ -1, & \text{Elbow down} \end{cases} \tag{5.40}$$

and can find the elbow point $\boldsymbol{P}_e$ by

$$\boldsymbol{P}_e = \frac{\boldsymbol{Q} + k_e\sqrt{\boldsymbol{Q}^2}}{-\boldsymbol{e}_\infty \cdot \boldsymbol{Q}} \tag{5.41}$$

This solution to the elbow up/elbow down configuration is geometrically intuitive, and easy to visualize.

Chapter 6

Curvature-based Descriptor

I read a book about anti-gravity.
It was hard to put down.

In this section we propose the curvature-based descriptor, which is a novel approach for 3D point cloud alignment. The descriptor is first described in [33]. It uses conformal geometric algebra to generate a descriptor. Section 9.5 and Section 9.6 presents the two papers published on the descriptor, and a detailed description will follow in this section.

6.1 Overview

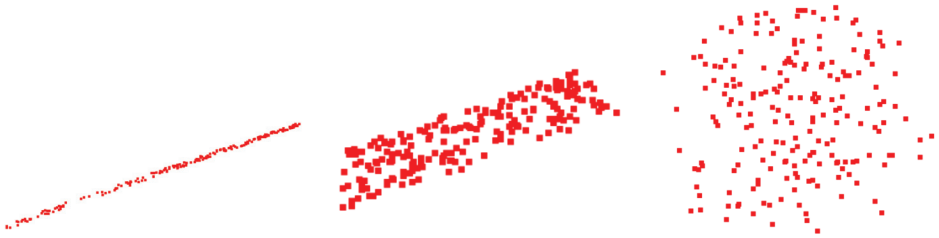
The curvature-based descriptor starts by first selecting keypoints in the point clouds $\boldsymbol{X}$ and $\boldsymbol{Y}$ based on the geometry of the neighbourhood of each point. This is done by using principal component analysis and shape factors. A descriptor is then calculated for each keypoint in $\boldsymbol{X}$ and $\boldsymbol{Y}$, where each descriptor consists of two spheres. Each sphere represents the local curvature around the point in two orthogonal directions. The point correspondence is established between these descriptors, which then is used to estimate the pose between the two point clouds.

6.2 Keypoint Extraction

For each point $\boldsymbol{p}_i \in \boldsymbol{X}$, the set of neighbouring points $\boldsymbol{N}_i$ is found using a radius search selection with a given distance r. As described in Section 2.6.1, radius search selection is preferred over a k-nearest neighbour scheme because it is independent of resolution.

A principal component analysis is performed on the set $\boldsymbol{N}_i$, so that the eigenvalues λ_1, λ_2 and λ_3, and the eigenvectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ can be found. The eigenvalues are used to determine the general distribution of the points in $\boldsymbol{N}_i$, while the eigenvectors are used to define the local reference frame.

Since the points are part of a surface, the general distribution of the points in $\boldsymbol{N}_i$ also defines the general shape of the surface. The shape of the surface can



(a) Point cloud with a shape factor $C_l \approx 1$ (b) Point cloud with a shape factor $C_p \approx 1$ (c) Point cloud with a shape factor $C_s \approx 1$

Figure 6.1: Examples of point clouds with different shape factors.

be analyzed using the eigenvalues of from the principal component analysis using *shape factors* [1, 44]

$$C_l = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3} \tag{6.1}$$

$$C_p = \frac{2(\lambda_2 - \lambda_3)}{\lambda_1 + \lambda_2 + \lambda_3} \tag{6.2}$$

$$C_s = \frac{3\lambda_3}{\lambda_1 + \lambda_2 + \lambda_3} \tag{6.3}$$

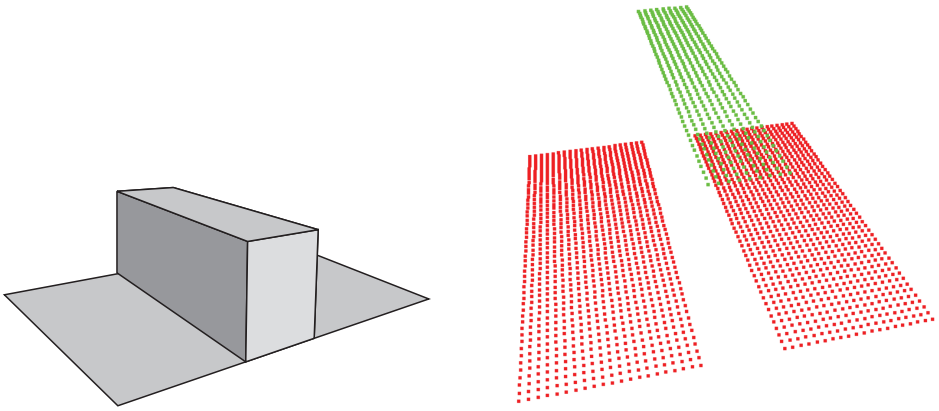
where $C_l + C_p + C_s = 1$. If C_l is dominant, then the surrounding point cloud surface is close to linear; if C_p is dominant, then the point cloud is close to being planar; while if C_s is dominant, than the surrounding point cloud is spherical or has a form with volume. An example of such point clouds can be seen in Figure 6.1.

A few assumptions can be made when analyzing shape factors of a point cloud, especially when the point cloud is taken with a 3D camera, or other projected point clouds. Firstly, since the points lie on a surface, the curvature of the surface has impact on the C_s shape factor. This is also true for corners, a property that is exploited in Harris Corner Detection [25] in 2D images. Secondly, the C_l shape factor only occurs if the point cloud has a ridge, as shown in Figure 6.2, or as a result of noise. This can be exploited if the model has characteristic ridge-like geometries, or to exclude noise patterns.

When all the shape factors are calculated and analyzed, the keypoints can be selected using the criteria

$$|\boldsymbol{N}_i| \geq n_{\min} \text{ and } (C_l \geq \delta_l \text{ or } C_p \geq \delta_p \text{ or } C_s \geq \delta_s) \tag{6.4}$$

where $n_{\min}$, δ_l, δ_p and δ_s are user-specified parameters. The parameter $n_{\min}$ specifies the minimum number of points in $\boldsymbol{N}_i$ that are required in order for the keypoint to be selected. This excludes the points that have a small neighbourhood, since these are most likely outlier points, and are more subjected to noise. δ_l, δ_p and δ_s are specified by the user, and they depend on the shape and form of the point cloud. The purpose is to select parameters so that unique points are selected, and that these are selected in both $\boldsymbol{X}$ and $\boldsymbol{Y}$. Having a low value of either δ_l, δ_p



(a) A drawn representation of a ridge shape.

(b) The point cloud representation of Figure 6.2a. Here, the green points are part of the ridge. This only occurs when the camera is viewing the ridge shape from the front.

Figure 6.2: An example of a surface where C_l is large.

or δ_s selects more points that have that specific characteristics, while high values chooses fewer points that are more specific. Choosing $\delta > 1$, effectively disregards that parameter when selecting keypoints. As seen in Figure 6.3, most of the model consists of flat cylindrical surfaces, which means a large number of points have a high C_p, while a low number of points have high C_l and C_s.

The selection of keypoints is done in both $\boldsymbol{X}$ and $\boldsymbol{Y}$, resulting in a set of keypoints, $\boldsymbol{X}_{\text{keypoints}}$ and $\boldsymbol{Y}_{\text{keypoints}}$ respectively.

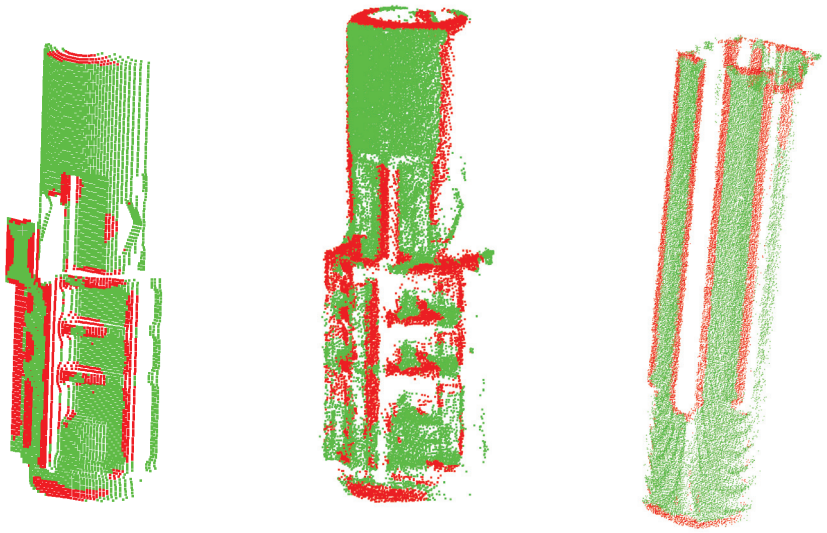
6.3 Sphere Fitting

The sphere fitting method presented in [19], which is explained in Chapter 5, is modified in the curvature-based descriptor by the introduction of weighting factors for the points, so that the objective function becomes

$$f(\boldsymbol{S}) = \sum_{i=1}^{n} w_i \frac{(\boldsymbol{S}^* \cdot \boldsymbol{P}_i)^2}{\boldsymbol{S}^{*2}} \tag{6.5}$$

This has the same effect as modifying the input point data of the minimization problem. The condition for optimality becomes $(\sum_{i=1}^{n} w_i \boldsymbol{P}_i(\boldsymbol{P}_i \cdot \boldsymbol{S}^*)) \wedge \boldsymbol{S}^* = 0$, which in matrix form is written

$$\boldsymbol{G} = \sum_{i=1}^{n} w_i([\boldsymbol{P}_i][\boldsymbol{P}_i]^{\mathrm{T}})\boldsymbol{M} \tag{6.6}$$



(a) A CAD model view of object A, with the approximately same view angle as in Figure 6.3b.

(b) A 3D camera measurement of object A

(c) A 3D camera measurement of object B

Figure 6.3: A sample of the keypoint selection process using (6.4) with the parameters $n = 200$, $\delta_l = 0.3$, $\delta_p = 1$ and $\delta_s = 0.3$. The red points are keypoints, while the green points are not. It is seen that the keypoints in Figure 6.3a and Figure 6.3b are similar, while that of Figure 6.3c is different. This is a desired behaviour as the match between the point cloud in Figure 6.3a and Figure 6.3b will be better than that of Figure 6.3a and Figure 6.3c.

where $[\boldsymbol{P}_i]$ is the column vector representation of the conformal geometric point $\boldsymbol{P}_i$, and

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix} \tag{6.7}$$

For each keypoint $\boldsymbol{p}_i$, weighting is introduced by defining the two planes

$$\boldsymbol{\Pi}_{i1} = \boldsymbol{P}_i \wedge \mathbf{v}_1 \wedge \mathbf{v}_3 \wedge \boldsymbol{e}_\infty, \qquad \boldsymbol{\Pi}_{i2} = \boldsymbol{P}_i \wedge \mathbf{v}_2 \wedge \mathbf{v}_3 \wedge \boldsymbol{e}_\infty \tag{6.8}$$

where $\boldsymbol{\Pi}_{i1}$ is the plane spanned by $\mathbf{v}_1$ and $\mathbf{v}_3$ through $\boldsymbol{P}_i$, and $\boldsymbol{\Pi}_{i2}$ is the plane spanned by $\mathbf{v}_2$ and $\mathbf{v}_3$ through $\boldsymbol{P}_i$. The two planes are used so that the curvature

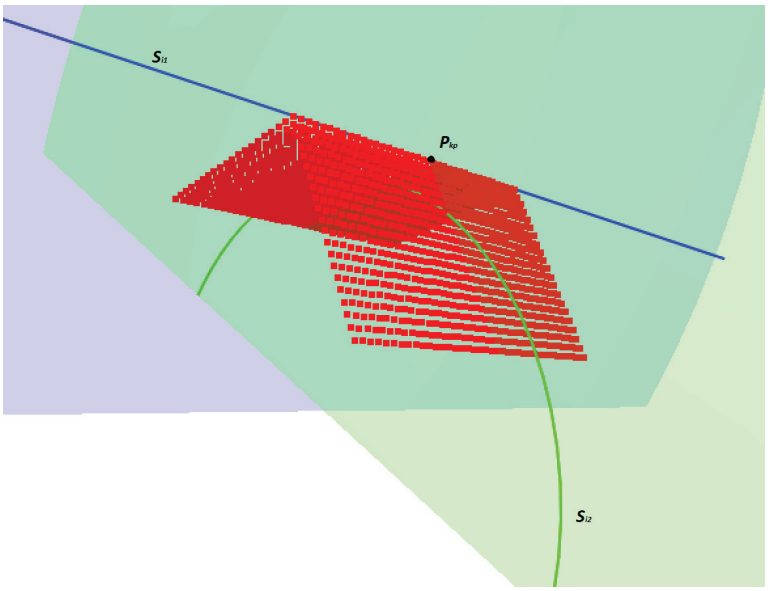


Figure 6.4: An example of the two spheres and the descriptor. The blue sphere, $\boldsymbol{S}_{i1}$, and green spheres, $\boldsymbol{S}_{i2}$, are illustrated as circles in the figure, for convenience. The green and blue planes are $\boldsymbol{\Pi}_{i1}$ and $\boldsymbol{\Pi}_{i2}$ respectively.

along the direction of $\mathbf{v}_1$ and $\mathbf{v}_2$ can be estimated from

$$\boldsymbol{G}_{\boldsymbol{P}_{i1}} = \sum_{i=0}^{n} e^{-\gamma|\boldsymbol{P}_i \cdot \boldsymbol{\Pi}^*_{i1}|}([\boldsymbol{P}_i][\boldsymbol{P}_i]^{\mathrm{T}})\boldsymbol{M} \tag{6.9}$$

$$\boldsymbol{G}_{\boldsymbol{P}_{i2}} = \sum_{i=0}^{n} e^{-\gamma|\boldsymbol{P}_i \cdot \boldsymbol{\Pi}^*_{i2}|}([\boldsymbol{P}_i][\boldsymbol{P}_i]^{\mathrm{T}})\boldsymbol{M} \tag{6.10}$$

where $|\boldsymbol{P}_i \cdot \boldsymbol{\Pi}^*_{ij}|$ is the distance from the point $\boldsymbol{p}_i$ to the plane $\boldsymbol{\Pi}_{ij}$, and $\gamma \in \mathbb{R}$ is a weighting parameter. It is seen that the weighting factor in $\boldsymbol{G}_{\boldsymbol{P}_{ij}}$ is unity when the point $\boldsymbol{p}_i$ is on the plane $\boldsymbol{\Pi}_{ij}$, and that the weight decreases when the distance from the plane to the point increases.

The two spheres

$$[\boldsymbol{S}^*_{i1}] = \alpha_{i1}\mathbf{v}_{i1*}, \qquad [\boldsymbol{S}^*_{i2}] = \alpha_{i2}\mathbf{v}_{i2*} \tag{6.11}$$

are then found from the eigenvector $\mathbf{v}_{ij*}$ corresponding to the smallest positive eigenvalue λ_{ij*} of the matrix $G_{\boldsymbol{P}_{ij}}$ for $j = 1, 2$. These two spheres are used to calculate the descriptor of $\boldsymbol{p}_i$. This is then repeated for all points in $\boldsymbol{X}_{\text{keypoints}}$ and $\boldsymbol{Y}_{\text{keypoints}}$. An example of the two spheres is shown in Figure 6.4.

As noted Section 3.2, there is no way to predictably construct a reference frame from the resulting orthogonal vectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$. The curvature-based descriptor goes around this, since the spheres do not depend on the directional vectors, but rather on the planes $\boldsymbol{\Pi}_{i1}$ and $\boldsymbol{\Pi}_{i2}$. This makes it possible to get the same result regardless of whether for instance $\mathbf{v}_i$ or $-\mathbf{v}_i$ is chosen.

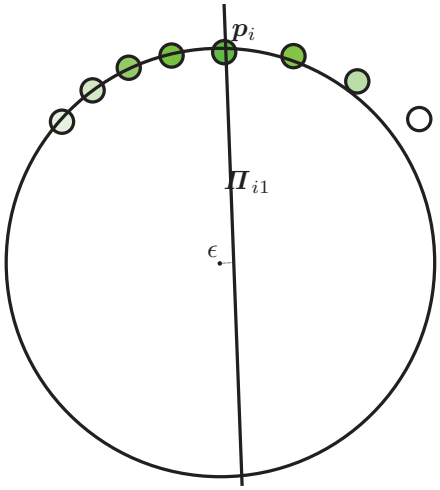


Figure 6.5: The green dots are points in the point cloud, and their shade represents their weight determined by the distance from the plane $\boldsymbol{\Pi}_{i1}$. The ϵ is the small bias that is generated because there are more points on the left side of $\boldsymbol{\Pi}_{i1}$, than on the right side.

6.3.1 Special Properties

When using the proposed sphere fitting method, the weighting factor weighs the points in such a manner that the closest points to the plane $\boldsymbol{\Pi}_{ij}$ are more significant than those far away. In regards to how projective geometry works, the resolution of a point cloud decreases the further it is from the camera. In some cases this means that for a given point $\boldsymbol{p}_i$ and its neighbouring points $\boldsymbol{N}_i$, there are more points on one side of the plane $\boldsymbol{\Pi}_{ij}$ than the other, as shown in Figure 6.5. This results in that the center of the sphere $\boldsymbol{S}_{ij}$ does not lie on the surface of $\boldsymbol{\Pi}_{ij}$. This bias is small, and does not seem to have a noticeable effect on the overall point correspondence, but it is a property worth exploring.

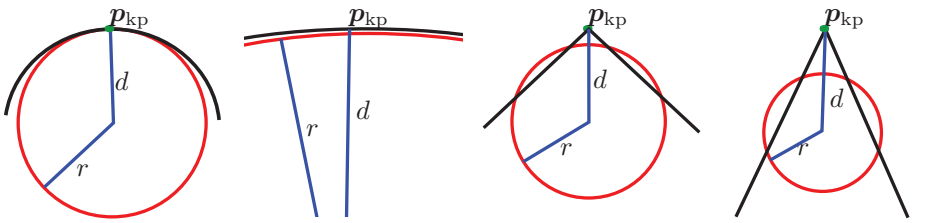
The weighting factor $e^{-\gamma|\boldsymbol{P}_i \cdot \boldsymbol{\Pi}_{ij}|}$ can be rewritten as a Gaussian distribution such that

$$\exp -\gamma|\boldsymbol{P}_i \cdot \boldsymbol{\Pi}_{ij}| = \exp -\frac{(x-\mu)^2}{2\sigma^2} \tag{6.12}$$

where

$$\gamma = \frac{1}{2\sigma^2}, \qquad |\boldsymbol{P}_i \cdot \boldsymbol{\Pi}_{ij}| = (x-\mu)^2 \tag{6.13}$$

which makes it easier to determine which points that should be chosen. In [33], it showed that for the specific case, the weight of $0.05 \leq \gamma \leq 0.2$ is the recommended, which is equal to a variance of $0.158 \leq \sigma \leq 0.316$. This means that if $\gamma = 0.05$, then points that are 0.316 mm away from the plane $\boldsymbol{\Pi}_{ij}$ has a weight factor of 0.05 or less.



(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. $r \ll d$.

Figure 6.6: A sample of resulting sphere estimates on different surfaces. The different values of r and d indicates what the different shapes are.

6.4 Point Correspondence Estimation

The point correspondence method of the descriptor was developed in [32]. For each keypoint, $\boldsymbol{p}_i$, and their estimated spheres, $\boldsymbol{S}_{i1}$ and $\boldsymbol{S}_{i2}$, a descriptor was calculated

$$\boldsymbol{F}_i = (\boldsymbol{S}_{i1} + \boldsymbol{S}_{i2}) \cdot \boldsymbol{P}_i \tag{6.14}$$

$$= \frac{1}{2}((d_1^2 - r_1^2) + (d_2^2 - r_2^2)) \tag{6.15}$$

$$\tag{6.16}$$

where d_j is the distance between between the center of the sphere $\boldsymbol{S}_{ij}$ and the point $\boldsymbol{p}_i$, and r_j is the radius of $\boldsymbol{S}_{ij}$ for $j = 1, 2$. This method was used due to the fact that in [33], the sphere fitting algorithms were very sensitive to noise when estimating planar or planar-like surfaces. When the point clouds had a planar shape, the radius was arbitrary large, making it hard to compare if two keypoints are equal based on radius alone.

It is the distance $\delta = d - r$, between the keypoint and the surface of the sphere which determines what shape the keypoint has, as seen in Figure 6.6. The distance δ can be approximated with the term $d^2 - r^2$ as shown above, which effectively cancels out the large r value, since the d value is also arbitrarily large when estimating a sphere.

To find the point correspondence we solve the minimization problem

$$\min_l g(\boldsymbol{F}_k, \boldsymbol{F}_l), \quad \forall\ \boldsymbol{F}_k \in \boldsymbol{X}_{\text{keypoints}}, \boldsymbol{F}_l \in \boldsymbol{Y}_{\text{keypoints}} \tag{6.17}$$

for each k where

$$g(\boldsymbol{F}_k, \boldsymbol{F}_l) = (\boldsymbol{F}_k - \boldsymbol{F}_l)^2 = \frac{1}{2}(\delta_{k1}^2 - \delta_{l1}^2 + \delta_{k2}^2 - \delta_{l2}^2)^2 \tag{6.18}$$

6.4.1 Special Cases

As seen with the special cases in Section 6.3, there are instances where this point correspondence method might not work optimally. This is because if the value of

$\boldsymbol{F}_i$ is negative and $\boldsymbol{F}_j$ is positive, then $g(\boldsymbol{F}_i, \boldsymbol{F}_j)$ is will have a minimum even if $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ are not corresponding.

6.5 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 the distance between each point in the point clouds $\boldsymbol{X}_{\text{keypoints}}$ and $\boldsymbol{Y}_{\text{keypoints}}$. This is straightforward, and can be done in the usual way using SVD as described in, e.g., [2, 57].

The pose estimation method used in the curvature-based descriptor is the motor estimation method presented in [58]. This was chosen because it was convenient to use with the other conformal geometric algebra methods in the descriptor.

Chapter 7

Experiments

What does a house wear?

Address

7.1 Setup

There has been conducted several experiments throughoutthe work of this thesis, and they focus on simulating a real world example of automated assembly.

The main setup is a scene which consists of multiple objects placed on a table, and a camera pointing towards the table and captures the scene.. Two robots are set up next to the table, and has a working area which covers the whole table. An overview can be seen in Figure 7.1.

The experiments mostly focused on the two car parts shown in Figure 7.2. These two parts had sufficiently unique shapes that they could be distinguished from each other, but also had similarities in their shape. They are constructed in such a way that they can be assembled into each other, and that there is only one possible solution, and that the margin for error is under 1 mm and 1°. This is a very likely scenario in an assembly situation, and also a very challenging task in computer vision. There were many more car parts to that were provided, but it was concluded that a solution that worked for the two parts in Figure 7.2 would also work for the other car parts with some parameter tweaked.

7.2 Camera

There were three different 3D cameras that were used in this thesis: Xtion PRO LIVE by Asus, Kinect v2 by Microsoft and Zivid by ZividLabs.

7.2.1 Asus XtionPRO LIVE

The Xtion PRO LIVE camera is an RGB-D camera with 640x480 resolution at 30 Hz. The camera uses structured lights to measure the distances, and works on a

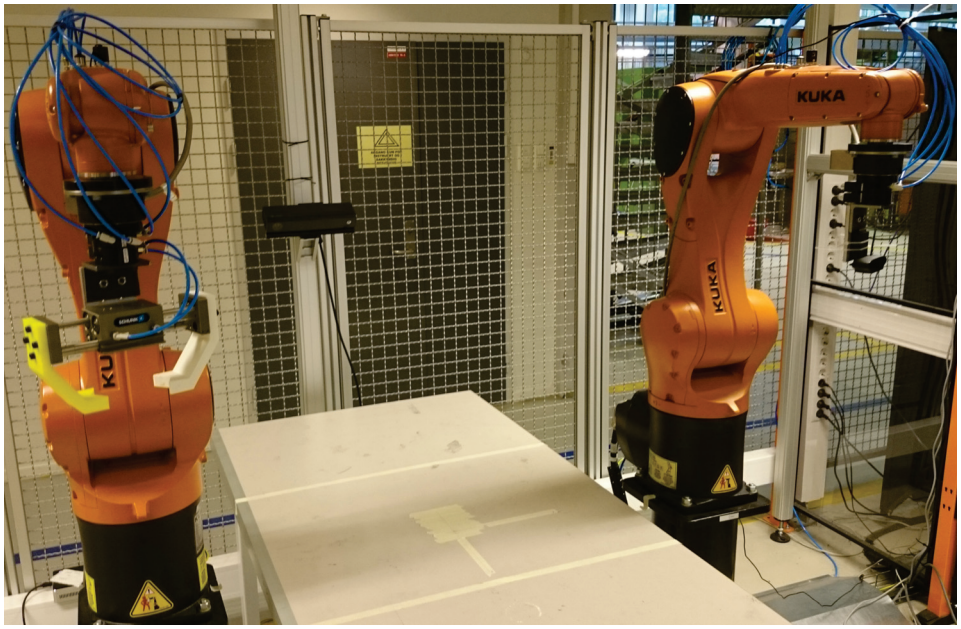
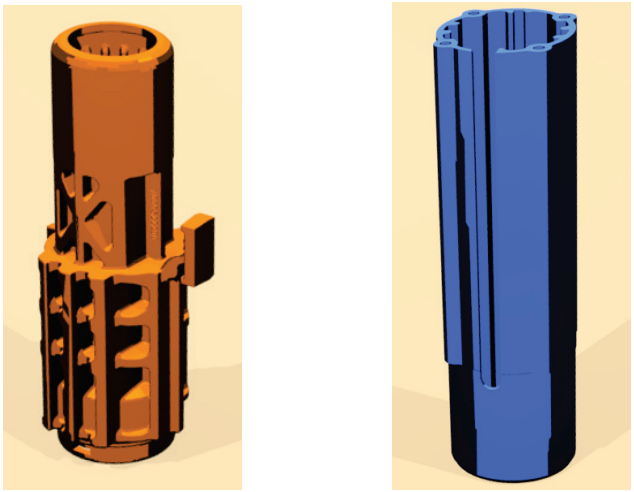


Figure 7.1: Overview of the Agilus robot cell. The two Agilus robots stand adjacent to each other, with a table between them. The camera is placed on the wall behind the table. In the image, the Kinect v2 is set up on the back wall.



(a) Car part A (b) Car part B

Figure 7.2: A 3D rendering of the two car parts used in the experiments.

range from 0.8 m to 3.5 m. The accuracy is not specified, but according to [43] the noise is measured to be $\pm 3.99 \frac{\text{mm}}{\text{m}}$.

This camera was the first 3D camera that was used, but the initial experiments showed that the accuracy and resolution was not sufficient to perform an accurate initial alignment.

7.2.2 Microsoft Kinect v2

The Kinect v2 camera is an RGB-D which uses time-of-flight to measure the distance. The resolution is 512x424, and has a range from 0.5 m to 4.5 m. The accuracy is not specified, but according to [43] the noise is measured to be $\pm 3.88 \frac{\text{mm}}{\text{m}}$.

This camera was used for most of the experiments. The accuracy was sufficient to perform initial alignment, and was used in the experiments in [55] and [30]. As shown in the experiments, the accuracy is not less than a millimeter, which makes it hard to achieve the specified accuracy with only this camera.

Though it is not documented, there were some initial experiments that showed that the distance measurements drifted depending on how long the camera was on. The camera was placed 1 m away from a wall and the camera took continuous measurements for 30 minutes, with over 20 000 measurements. The measurements showed that the measurements changed over the course of the experiment, and varied with about 1.5 cm. This is because the components inside the camera are heated up over time, which changed the characteristics of the camera.

7.2.3 ZividLabs Zivid 3D Camera

The Zivid camera is a RGB-D camera which uses light projection to measure the distances. The camera has a resolution of 1920x1200, and has a range from 0.6 m to 1.1 m. The accuracy of the camera is 0.1 mm at a distance of 0.6 mm.

The camera was used in [32] and [33], and the results show that it is possible to achieve the sub-millimeter accuracy that is required for the assembly operation.

7.3 Point Clouds

There were two main methods of acquiring point clouds in the conducted experiments: Viewpoint sampling on the CAD models, and 3D camera capturing on the physical objects.

Viewpoint sampling has been discussed in Section 2.6 and is where a 3D CAD model is rendered in a 3D image from different viewpoints using a virtual 3D camera. In the experiments, the tessellated sphere module from the PCL library [49] was used. The module first sets the CAD model in the origin of a 3D space, then a sphere is created so that it encapsulates the model. This sphere is divided into polyhedrons, as shown in Figure 7.3, where a virtual camera is placed in each corner of the polyhedron, pointing towards the origin. In the experiments, a sphere constructed from 80 triangles where used, resulting in 42 corners, which resulted in 42 viewpoint point clouds per CAD model. The resolution of these images depended on the experiment, and ranged from 90×90 pixels to 400×400 pixels.

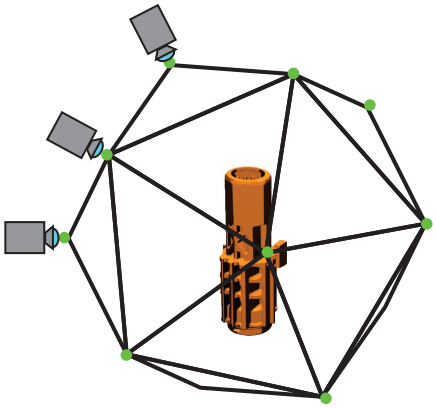


Figure 7.3: A tessellated sphere surrounding a CAD model. Each of the green dots is a viewpoint position for the virtual camera.

Figure 7.4: The Kuka KR6 sixx R900 Agilus robot.

The point clouds captured with a 3D camera came either from the XtionPRO LIVE, the Kinect v2 or the Zivid camera. The scene was often objects on a table, and the points that were not part of the objects were mostly removed, either by hand or with the algorithm shown in [55].

7.4 Robots

The two robots that are used in the experiments are the Kuka KR6 sixx R900, also known as the Kuka Agilus. The robots are 6DOF, and have a maximum payload of 6 kg, and a total reach of 901.5 mm. An image is shown in Figure 7.4.

The robots were set up with different grippers and camera brackets, depending on the experiments. These were custom made either in plastic or aluminum.

On both robots there were robotic tool changers and a 6DOF force/torque sensor on the end-effector, but these were never used in any of the experiments.

Chapter 8

Conclusion

What do you call a dog magician?

A Labracadabrador

The thesis presented methods for pose alignment and point correspondence estimation of 3D point clouds and inverse kinematics of robot arms. The approaches proposed in this thesis are based on conformal geometric algebra, an extension of Euclidean algebra.

The thesis presented the Curvature-Based Descriptor, a novel approach for the initial alignment between two point clouds. The curvature-based descriptor is a descriptor which describes the local curvature around a point in the point cloud. The local curvature is expressed with two spheres generated using conformal geometric algebra. The thesis also presented the preprocessing steps which are used to segment the point cloud using RANSAC, and a keypoint extraction using shape factors method which extracts certain points from the point cloud, making the point correspondence more accurate.

The inverse kinematics presented in this thesis is an analytic solution which uses conformal geometric algebra. The solution was presented for the Kuka KR6 R900 sixx robot and the Universal Robots UR5 robot. All singularities and all configurations were accounted for in the solutions.

The thesis has presented several experimental results. These experiments show the results from various methods performing point cloud alignment. The results show that it is possible to achieve a sub-millimeter accuracy for position estimation of an object using FPFH as an initial alignment, followed by performing SIFT on an image taken from a 2D camera attached to a robot arm.

The thesis has also showed through experiments that the curvature-based alignment method, after applying the preprocessing, achieve a sub-millimeter accuracy on its own, an accuracy that is not achieved with any of the other 3D alignment methods.

8.1 Future Work

An experiment that was not published when using the curvature-based description showed an interesting result. When using two of the exact same point cloud with a known displacement between each other, we chose a set of random point as keypoints. The same points were chosen from both point clouds. This gave a 100 % accurate point correspondence. Our thought is that this is because the points are random, they are not likely to be similar to each other. It would be interesting to mimic this behaviour in the keypoint extraction method.

There have been two point correspondence methods that have been presented in this thesis. One uses the r and d parameters separately to achieve correspondence, while the other uses the difference between r and d. Both have their advantages, and should be combined in some form so that they can exploit both advantages.

When selecting a neighbourhood for the shape factors, the proposed method use principal component analysis. Principal component analysis moves the centroid to the origin in order to calculate the principal axes, however, in [44] they move the keypoint itself to the origin. It is shown in the thesis that 3D cameras has projective geometry, and that this creates a bias to one side. This bias follows when using centroid, but not when using the selected point. It would be interesting to see if this has any effect on the performance.

Chapter 9

Publications

How do trees get online?

They log in.

The following publications come in chronological order. In this section, each paper is lightly explained, focusing more on the reasoning behind the paper. In the sections following, the papers are explained in a summary form.

Paper 1 presents inverse kinematics using conformal geometric algebra. When we first started with inverse kinematics and conformal geometric algebra, there were examples of its use, such as in [27, 40]. These publications focus on inverse kinematics in graphic animations, and therefore do not have a full 6DOF solution, and had not taken singularities and configurations into account to a degree that they could be implemented on an industrial robot. The paper therefore presents such a solution.

Paper 2 presents an object detection algorithm which uses conformal geometric algebra and RANSAC. At the time, we were researching alignment of 3D point clouds. A problem we faced was that the point cloud captured by the 3D camera covered a larger surface than that of the objects in the 3D CAD model. The 3D camera captured the table and the surrounding background as well. We therefore wanted to use segmentation to select only the objects that were on the table by estimating the plane of the table and selecting only the points on top of it.

Aksel Sveier was working on RANSAC with conformal geometric algebra at the time, and through our combined research we found that this could be used as a object detection algorithm.

Paper 3 presents an alignment method which achieves sub-millimeter accuracy. Throughout our research, we found that many of the algorithms for 2D and 3D alignment were limited in their accuracy. Sub-millimeter accuracy was crucial for the assembly process, so Asgeir Bjørkedal and Kristoffer Larsen started to research if we could combine the different methods in order to achieve the necessary accuracy.

We therefore developed a method which combined a stationary 3D camera with a 2D camera attached to a robot arm. The method achieved the necessary accuracy for an automated assembly.

Paper 4 presents an initial alignment for point clouds which uses conformal geometric algebra. The initial idea was to use the least-square optimization method presented in [59], and that instead of using a point correspondence, we could use a correspondence between different geometric objects. We therefore developed a global descriptor method that used geometric algebra.

Paper 5 presents the curvature-based descriptor, which is described in Chapter 6. The method is based on the experiences from Paper 4, and was developed with sub-millimeter accuracy as a goal. The experiments show that the descriptor performs better than several state-of-the-art methods for estimating the position of the objects.

Paper 6 presents an improvement to the curvature-based descriptor. In Paper 5, the point correspondence method used the of the radius and the distance between the center of the sphere and the keypoint separately, which in some cases was unstable. The improvement addresses this, and achieves a better point correspondence and an even better position estimation.

9.1 Paper 1: Inverse Kinematics for Industrial Robots using Conformal Geometric Algebra

By Adam Leon Kleppe and Olav Egeland

This paper presents inverse kinematics on the Kuka Agilus KR6R900 and the Univeral Robot UR5 using conformal geometric algebra.

This paper shows a precise example of how conformal geometric algebra can be used in inverse kinematics, and goes into details such as defining specific configurations and defining singularities.

Modeling, Identification and Control, Vol. 37, No. 1, 2016, pp. 63–75, ISSN 1890–1328

Inverse Kinematics for Industrial Robots using Conformal Geometric Algebra

A. Kleppe [1] O. Egeland [1]

[1] *Department of Production and Quality Engineering, Norwegian University of Science and Technology, N-7491 Trondheim, Norway. E-mail:* {adam.l.kleppe,olav.egeland}@ntnu.no

Abstract

This paper shows how the recently developed formulation of conformal geometric algebra can be used for analytic inverse kinematics of two six-link industrial manipulators with revolute joints. The paper demonstrates that the solution of the inverse kinematics in this framework relies on the intersection of geometric objects like lines, circles, planes and spheres, which provides the developer with valuable geometric intuition about the problem. It is believed that this will be very useful for new robot geometries and other mechanisms like cranes and topside drilling equipment. The paper extends previous results on inverse kinematics using conformal geometric algebra by providing consistent solutions for the joint angles for the different configurations depending on shoulder left or right, elbow up or down, and wrist flipped or not. Moreover, it is shown how to relate the solution to the Denavit-Hartenberg parameters of the robot. The solutions have been successfully implemented and tested extensively over the whole workspace of the manipulators.

Keywords: Conformal Geometric Algebra, Inverse Kinematics, Agilus sixx R900, UR5

1 Introduction

Analytical inverse kinematics is a well-developed problem in robotics. Solutions are available as text-book material for revolute robots with a spherical wrist, or with three consecutive parallel axes [Siciliano et al. (2009); Spong et al. (2006)]. The solutions are given in terms of trigonometric expressions, which are straightforward to find, although they can be somewhat involved. The complexity of the equations is partly related to the book-keeping of the different solutions related to shoulder left or right, elbow up or down, and wrist flipped or not.

The recently developed formulation of conformal geometric algebra as presented in [Dorst et al. (2009); Hildenbrand (2013); Perwass (2009)] provides additional insight into the problem. This formulation has very efficient tools to define geometric objects in the form of lines, circles, planes and spheres, and includes the geometric product, which is used to calculate intersections of such geometric objects and the distance between different objects. The formulation extends the 3-dimensional Euclidean space with 2 extra dimensions resulting in a homogeneous space including the point at infinity. In this formalism, the inverse kinematics has been previously solved for a robot with 5 revolute joints in terms of spheres, planes and lines, and the intersection of these geometric objects [Hildenbrand (2013); Hildenbrand et al. (2005); Hildenbrand et al. (2006); Zamora and Bayro-Corrochano (2004)]. These inverse kinematic solutions have primarily been developed for graphical rendering, as the focus has been on the link configurations, whereas the joint angles are only given in terms of the cosines of the angles, which means that there is no systematic way of determining the right quadrant of the joint angles. Still, this work clearly demonstrates that the conformal geometric algebra is a very powerful tool for inverse kinematics, which makes

doi:10.4173/mic.2016.1.6

it interesting to explore this formulation more in detail to investigate how it can be employed to solve and implement a range of practical kinematic problems in robotics. To do this we revisit the well-established inverse kinematic problem for robots to demonstrate how conformal geometric algebra can be used in robotics.

In this work we extend the existing solutions for analytic inverse kinematics based on conformal geometric algebra to obtain a systematic way of calculating the signs and quadrants of the joint angles. This includes the calculation of consistent solutions corresponding to shoulder left and right, elbow up or down and wrist flipped or not. Moreover, it is shown how the rotational direction of the joint angles are related to the Denavit-Hartenberg parameters. It is also shown how to describe links that have both a and d translations in the Denavit-Hartenberg convention. The proposed method is implemented for the Agilus R900 Sixx robot, which is a 6 DOF robot with a spherical wrist, and the UR5, which is a 6 DOF robot with parallel axes for joints 2, 3 and 4. Also singularities are discussed, and it is explained how the singularities appear in the solution based on conformal geometric algebra.

The paper is organized as follows. First a brief presentation of manipulator kinematics is given. Then the basics of conformal geometric algebra is presented, which includes a discussion on how to determine the sign of rotation in this formulation. Then the implementation of the analytic inverse kinematics is presented for the Agilus R900 Sixx and the UR5 robot.

2 Manipulator kinematics

The Denavit-Hartenberg convention is commonly used for describing robot kinematics. The convention describes the link transformation in terms the homogeneous link transformation matrix

$$\boldsymbol{T}_{(i,i-1)} = \mathrm{Rot}_{z,\theta_i}\mathrm{Trans}_{z,d_i}\mathrm{Trans}_{x,a_i}\mathrm{Rot}_{x,\alpha_i} \tag{1}$$

This can be used to calculate the forward kinematics

$$\boldsymbol{T}_{06} = \begin{pmatrix} \boldsymbol{n}_e & \boldsymbol{s}_e & \boldsymbol{a}_e & \boldsymbol{p}_e \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

of a robot with six links as

$$\boldsymbol{T}_{06} = \boldsymbol{T}_{01}\boldsymbol{T}_{12}\boldsymbol{T}_{23}\boldsymbol{T}_{34}\boldsymbol{T}_{45}\boldsymbol{T}_{56} \tag{3}$$

The Denavit-Hartenberg parameters for the Aguilus robot are presented in Table 1 and Figure 2b, while the Denavit-Hartenberg parameters for the UR5 robot are shown Table 2 and Figure 6b.

Link	θ_i[rad]	d_i[mm]	a_i[mm]	α_i[rad]
1	θ_1	-400	25	$\frac{\pi}{2}$
2	θ_2	0	455	0
3	$\theta_3 - \frac{\pi}{2}$	0	35	$\frac{\pi}{2}$
4	$\theta_4 + \frac{\pi}{2}$	-420	0	$-\frac{\pi}{2}$
5	$\theta_5 - \frac{\pi}{2}$	0	0	$\frac{\pi}{2}$
6	θ_6	-80	0	π

Table 1: DH-table for the Agilus R900 sixx robot

Link	θ_i[rad]	d_i[mm]	a_i[mm]	α_i[rad]
1	θ_1	89.2	0	$\frac{\pi}{2}$
2	θ_2	0	-425	0
3	$\theta_3 - \frac{\pi}{2}$	0	-392.43	0
4	θ_4	109.15	0	$\frac{\pi}{2}$
5	$\theta_5 - \frac{\pi}{2}$	94.65	0	$-\frac{\pi}{2}$
6	θ_6	82.3	0	0

Table 2: DH-table for the UR5 robot

3 Conformal Geometric Algebra

In this paper conformal geometric algebra is used for the inverse kinematics of robots. The main difference to the usual geometric formulation used in robotics is the introduction of the geometric product, and the extension of the 3 dimensional Euclidean space with 2 additional dimensions. This provides us with some very efficient tools, in particular, the formulation makes it very simple to define geometric objects in the form of lines, planes, circles and spheres. In addition, it is easy to calculate the occurrence of intersections between the geometric objects, and the distance between objects.

The Euclidean space $\mathbb{R}^3$ is described with the orthogonal unit vectors e_1, e_2, e_3. The vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ in Euclidean space are given by $\boldsymbol{a} = a_1e_1 + a_2e_2 + a_3e_3$ and $\boldsymbol{b} = b_1e_1 + b_2e_2 + b_3e_3$. The geometric product is defined as

$$\boldsymbol{ab} = \boldsymbol{a}\cdot\boldsymbol{b} + \boldsymbol{a}\wedge\boldsymbol{b} \tag{4}$$

where $\boldsymbol{a}\cdot\boldsymbol{b} = a_1b_1 + a_2b_2 + a_3b_3$ is the inner product, which is a scalar, and

$$\begin{aligned}\boldsymbol{a}\wedge\boldsymbol{b} =& (a_1b_2 - a_2b_1)e_1e_2 + (a_2b_3 - a_3b_2)e_2e_3 \\ &+ (a_3b_1 - a_1b_3)e_3e_1\end{aligned} \tag{5}$$

is the outer product, which is a bivector, as it is the sum of terms including the bivectors e_2e_3, e_3e_1 and e_1e_2. It is noted that $\boldsymbol{a}\cdot\boldsymbol{b} = \boldsymbol{b}\cdot\boldsymbol{a}$, and $\boldsymbol{a}\wedge\boldsymbol{b} = -\boldsymbol{b}\wedge\boldsymbol{a}$, and that

$$e_ie_j = e_i\cdot e_j + e_i\wedge e_j = \begin{cases} 1, & i = j \\ e_i\wedge e_j, & i \neq j \end{cases}$$

where $e_ie_j = e_i\wedge e_j = -e_j\wedge e_i = -e_je_i$ whenever $i \neq j$. It follows that $\boldsymbol{a}\wedge\boldsymbol{a} = 0$.

The 5-dimensional conformal space is obtained by extending the 3-dimensional Euclidean space with 2 orthogonal dimensions with basis vectors e_+ and e_- so that $e_+ \cdot e_+ = 1$ and $e_- \cdot e_- = -1$. A change of basis is done with $e_\infty = e_- + e_+$ and $e_0 = (1/2)(e_- - e_+)$, which implies that $e_\infty \cdot e_\infty = e_0 \cdot e_0 = 0$ and $e_\infty \cdot e_0 = -1$.

3.1 Multivectors

A multivector $\boldsymbol{a}$ in Euclidean space is a linear combination of the basis elements

$$\{1, e_1, e_2, e_3, e_2e_3, e_3e_1, e_1e_2, e_1e_2e_3\}$$

A multivector $\boldsymbol{A}$ in conformal space is a linear combination of the basis elements

$$\{1, e_0, e_1, e_2, e_3, e_\infty, e_0e_1, \ldots, e_0e_1e_2e_3e_\infty\}$$

The geometric product of two multivectors $\boldsymbol{A}$ and $\boldsymbol{B}$ is given by

$$\boldsymbol{AB} = \boldsymbol{A} \cdot \boldsymbol{B} + \boldsymbol{A} \wedge \boldsymbol{B}$$

The outer product has the property that $\boldsymbol{A} \wedge \boldsymbol{A} = 0$ for any multivector $\boldsymbol{A}$.

3.2 Duals and the pseudoscalar

The pseudoscalar in the Euclidean space $\mathbb{R}^3$ is $\boldsymbol{I}_E = e_1e_2e_3$. The Euclidean dual of a multivector $\boldsymbol{a}$ is

$$\boldsymbol{a}^+ = \boldsymbol{a}\boldsymbol{I}_E^{-1}, \quad \boldsymbol{I}_E^{-1} = e_3e_2e_1 \tag{6}$$

The square of the Euclidean pseudoscalar is $\boldsymbol{I}_E^2 = -1$, and it follows that the dual of the Euclidean dual is $(\boldsymbol{a}^+)^+ = -\boldsymbol{a}$.

The conformal pseudoscalar is $\boldsymbol{I}_c = e_0\boldsymbol{I}_E e_\infty$. The conformal dual of a multivector $\boldsymbol{A}$ in conformal space is

$$\boldsymbol{A}^* = \boldsymbol{A}\boldsymbol{I}_c^{-1}, \quad \boldsymbol{I}_c^{-1} = e_0\boldsymbol{I}_E^{-1}e_\infty \tag{7}$$

As in the Euclidean case, the square of the pseudoscalar is $\boldsymbol{I}_c^2 = -1$, and it follows that the dual of the dual is $(\boldsymbol{A}^*)^* = -\boldsymbol{A}$.

3.3 Conformal representation of Euclidean objects

In this paper the representation of geometric objects and their duals is based on the formulation in [Dorst et al. (2009)]. It is noted that an alternative formulation is presented in [Hildenbrand (2013)], where the direct form of [Dorst et al. (2009)] is presented as the dual form.

The Euclidean point $\boldsymbol{p}$ is represented in conformal space by the multivector

$$\boldsymbol{P} = C(\boldsymbol{p}) = \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^2 e_\infty + e_0$$

Starting from the representation of a point in conformal space the direct representation in conformal space of several Euclidean geometric objects can be generated with the outer product.

Let $\boldsymbol{P}_A$, $\boldsymbol{P}_B$ and $\boldsymbol{P}_C$ be the conformal representation of the points on a circle in Euclidean space. The direct representation of the circle in conformal space is then

$$\boldsymbol{C} = \boldsymbol{P}_A \wedge \boldsymbol{P}_B \wedge \boldsymbol{P}_C$$

A line in Euclidean space has the direct conformal representation

$$\boldsymbol{L} = \boldsymbol{P}_A \wedge \boldsymbol{P}_B \wedge e_\infty$$

where $\boldsymbol{P}_A$ and $\boldsymbol{P}_B$ are the conformal representation of two points on the line. A sphere in Euclidean space has the direct conformal representation

$$\boldsymbol{S} = \boldsymbol{P}_A \wedge \boldsymbol{P}_B \wedge \boldsymbol{P}_C \wedge \boldsymbol{P}_D$$

where $\boldsymbol{P}_A$, $\boldsymbol{P}_B$, $\boldsymbol{P}_C$ and $\boldsymbol{P}_D$ are conformal representations of points on the sphere that are not all in the same plane. A plane in Euclidean space has the direct conformal representation

$$\boldsymbol{\Pi} = \boldsymbol{P}_A \wedge \boldsymbol{P}_B \wedge \boldsymbol{P}_C \wedge e_\infty$$

where $\boldsymbol{P}_A$, $\boldsymbol{P}_B$ and $\boldsymbol{P}_C$ are conformal representations of points on the plane that are not collinear. In addition, the points $\boldsymbol{P}_A$ and $\boldsymbol{P}_B$ constitute a point pair

$$\boldsymbol{Q} = \boldsymbol{P}_A \wedge \boldsymbol{P}_B$$

A sphere $\boldsymbol{S}$ has the dual form

$$\boldsymbol{S}^* = \boldsymbol{P} - \frac{1}{2}\rho^2 e_\infty \tag{8}$$

where $\boldsymbol{P}$ center point and ρ is the radius of the sphere in Euclidean space. A plane $\boldsymbol{\Pi}$ has the dual form

$$\boldsymbol{\Pi}^* = \boldsymbol{n} + de_\infty \tag{9}$$

where $\boldsymbol{n}$ is the normal vector of the plane in Euclidean space and d is the distance from the origin.

3.4 Intersections

The intersection or meet $\boldsymbol{M}$ of two geometric objects $\boldsymbol{A}$ and $\boldsymbol{B}$ represented in the direct form in conformal space is given in terms of the dual $\boldsymbol{M}^* = \boldsymbol{A}^* \wedge \boldsymbol{B}^*$, or, equivalently, in the direct form as $\boldsymbol{M} = \boldsymbol{A}^* \cdot \boldsymbol{B}$. It is noted that the intersection of two planes $\boldsymbol{\Pi}_1$ and $\boldsymbol{\Pi}_2$ is the dual line $\boldsymbol{L}^* = \boldsymbol{\Pi}_1^* \wedge \boldsymbol{\Pi}_2^*$, the intersection of a plane $\boldsymbol{\Pi}$ and a sphere $\boldsymbol{S}$ is the dual circle $\boldsymbol{C}^* = \boldsymbol{\Pi}^* \wedge \boldsymbol{S}^*$, and the intersection of a plane $\boldsymbol{\Pi}$ and a circle $\boldsymbol{C}$ is a dual point pair $\boldsymbol{Q}^* = \boldsymbol{\Pi}^* \wedge \boldsymbol{C}^*$.

3.5 Distances

The distance between geometric objects is related to the inner product in some cases. The Euclidean distance d from a point $\boldsymbol{P}$ to a plane $\boldsymbol{\Pi}$ is given by the inner product in conformal space as $d = -\boldsymbol{P} \cdot \boldsymbol{\Pi}^*$ where d is positive if the point is in the direction of the normal vector. The Euclidean distance d between two points represented by $\boldsymbol{P}_A$ and $\boldsymbol{P}_B$ is given by $d^2 = -2\boldsymbol{P}_A \cdot \boldsymbol{P}_B$.

3.6 Horizon calculation

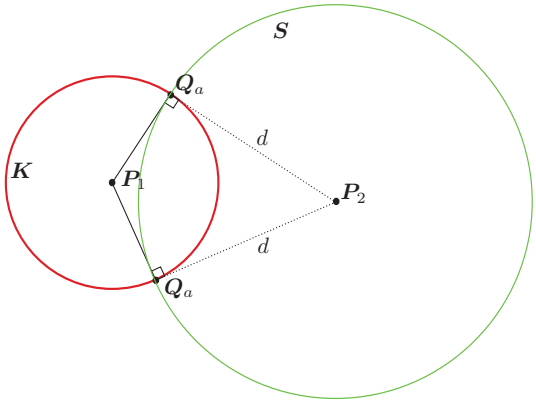


Figure 1: The point pair $\boldsymbol{Q}_a$ are two points which are d away from $\boldsymbol{P}_2$ and 90° between $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$

Consider two points with conformal representations $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$. Suppose that the two points are connected with a link with a 90° offset of length d. Then the offset can be located with the horizon technique presented in [Hildenbrand (2013)]. First the dual sphere $\boldsymbol{S}^* = \boldsymbol{P}_2 - \frac{1}{2}d^2 e_\infty$ with center point $\boldsymbol{P}_2$ and radius d is defined. Next, define the sphere $\boldsymbol{K}^* = \boldsymbol{P}_1 - (\boldsymbol{P}_1 \cdot \boldsymbol{S}^*)e_\infty$ with center in $\boldsymbol{P}_1$. Then the intersection of the spheres $\boldsymbol{S}$ and $\boldsymbol{K}$ will be the horizon defined by the circle

$$\boldsymbol{C}^* = \boldsymbol{K}^* \wedge \boldsymbol{S}^* \tag{10}$$

This circle is the set of all points with a 90° offset of length d. The intersection of this circle with a plane $\boldsymbol{\Pi}$ that contains both $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ will give a point pair $\boldsymbol{Q} = \boldsymbol{C}^* \cdot \boldsymbol{\Pi}$ where the two points of the point pair are on the tangent line from the point $\boldsymbol{P}_1$ with an offset d from $\boldsymbol{P}_2$.

An example of this can be seen in Figure 1.

3.7 Calculation of angles

In this section it is shown how to calculate the angle of rotation between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, and how to define the sign of the angle according to a defined direction of rotation. The corresponding unit vectors are given by $\hat{\boldsymbol{a}} = \boldsymbol{a}/\|\boldsymbol{a}\|$ and $\hat{\boldsymbol{b}} = \boldsymbol{b}/\|\boldsymbol{b}\|$. The geometric product of $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{b}}$ is

$$\hat{\boldsymbol{a}}\hat{\boldsymbol{b}} = \hat{\boldsymbol{a}} \cdot \hat{\boldsymbol{b}} + \hat{\boldsymbol{a}} \wedge \hat{\boldsymbol{b}} \tag{11}$$

The inner product of the two Euclidean unit vectors $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{b}}$ is the usual scalar product, which means that

$$\hat{\boldsymbol{a}} \cdot \hat{\boldsymbol{b}} = \cos\theta \tag{12}$$

where θ is the angle between the vectors. The outer product is

$$\hat{\boldsymbol{a}} \wedge \hat{\boldsymbol{b}} = \sin\theta \hat{\boldsymbol{N}} \tag{13}$$

where

$$\hat{\boldsymbol{N}} = \pm \frac{\hat{\boldsymbol{a}} \wedge \hat{\boldsymbol{b}}}{\left\|\hat{\boldsymbol{a}} \wedge \hat{\boldsymbol{b}}\right\|} \tag{14}$$

is a unit bivector that defines the plane of rotation from $\hat{\boldsymbol{a}}$ to $\hat{\boldsymbol{b}}$. The plus applies if the rotation from $\hat{\boldsymbol{a}}$ to $\hat{\boldsymbol{b}}$ is counter-clockwise in the plane defined by $\hat{\boldsymbol{N}}$, while the minus applies if the rotation in clockwise.

Equations 13 and 14 give the following expressions for the sine and cosine of the angle:

$$\begin{aligned} \cos\theta &= \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\|\boldsymbol{a}\| \, \|\boldsymbol{b}\|} \\ \sin\theta &= \frac{\boldsymbol{a} \wedge \boldsymbol{b}}{\|\boldsymbol{a}\| \, \|\boldsymbol{b}\|} \hat{\boldsymbol{N}}^{-1} \end{aligned} \tag{15}$$

where

$$\hat{\boldsymbol{N}}^{-1} = \pm \frac{\hat{\boldsymbol{b}} \wedge \hat{\boldsymbol{a}}}{\left\|\hat{\boldsymbol{b}} \wedge \hat{\boldsymbol{a}}\right\|} \tag{16}$$

is the inverse of $\hat{\boldsymbol{N}}$, which is equal to the reverse bivector.

It follows that the angle θ can be computed from

$$\theta = \operatorname{Atan2}\left[(\boldsymbol{a} \wedge \boldsymbol{b})\, \hat{\boldsymbol{N}}^{-1}, \boldsymbol{a} \cdot \boldsymbol{b}\right] \tag{17}$$

This approach ensures that the angle is calculated with the right sign.

In the inverse kinematics problem the two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ will typically be directional vectors of a line, or the normal vector of a plane. The directional vector of a line $\boldsymbol{L}$ is computed from

$$(\boldsymbol{L} \cdot e_0) \cdot e_\infty \tag{18}$$

while the normal vector of a plane $\boldsymbol{\Pi}$ is computed from

$$-(\boldsymbol{\Pi}^* \wedge e_\infty) \cdot e_0 \tag{19}$$

The rotation plane perpendicular to a line $\boldsymbol{L}$ is found from

$$\hat{\boldsymbol{N}} = -\frac{(\boldsymbol{L}^* \wedge e_\infty) \cdot e_0}{\|(\boldsymbol{L}^* \wedge e_\infty) \cdot e_0\|} \tag{20}$$

while the rotation plane parallel to a plane $\mathbf{\Pi}$ can be calculated from

$$\hat{\boldsymbol{N}} = -\frac{(\mathbf{\Pi} \cdot e_0) \cdot e_\infty}{\|(\mathbf{\Pi} \cdot e_0) \cdot e_\infty\|} \tag{21}$$

Note that the sign of the rotation plane $\hat{\boldsymbol{N}}$ for a robot joint must be selected so that the sign of the rotation is correct. This will be the case if the rotation axis $\boldsymbol{z}$ of the Denavit-Hartenberg convention is the Euclidean dual of the rotation plane, that is,

$$\hat{\boldsymbol{N}}^* = \boldsymbol{z} \tag{22}$$

4 Inverse Kinematics of the Agilus sixx R900 robot

The input parameters to the inverse kinematics are the position vector $\boldsymbol{p}_e$, the approach vector $\boldsymbol{a}_e$, the slide vector $\boldsymbol{s}_e$ and the normal vector $\boldsymbol{n}_e$ of the end-effector. Then the conformal representations of $\boldsymbol{p}_e$ and the wrist position $\boldsymbol{p}_e + d_6\boldsymbol{a}_e$ are given by

$$\boldsymbol{P}_e = C(\boldsymbol{p}_e) \tag{23}$$
$$\boldsymbol{P}_w = C(\boldsymbol{p}_e + d_6\boldsymbol{a}_e) \tag{24}$$

where d_6 is the distance between the end-effector and the wrist, which for the Agilus is 80 mm, as shown in Table 1.

The vertical plane $\mathbf{\Pi}_c$, which is the cross section of the robot through the wrist point, is then defined by

$$\mathbf{\Pi}_c = e_0 \wedge e_3 \wedge \boldsymbol{P}_w \wedge e_\infty \tag{25}$$

We define three configurations: Front/Back, which defines if it is the front or back of the robot that faces the end-effector; Elbow up/Elbow down, which defines if the elbow joint is up or down; and Flip/No Flip, which defines if the wrist joint is flipped or not.

These configurations are selected with the following parameters:

$$k_{fb} = \begin{cases} 1 & \text{if front} \\ -1 & \text{if back} \end{cases} \tag{26}$$

$$k_{ud} = \begin{cases} 1 & \text{if elbow up} \\ -1 & \text{if elbow down} \end{cases} \tag{27}$$

$$k_{fn} = \begin{cases} 1 & \text{if flip} \\ -1 & \text{if no flip} \end{cases} \tag{28}$$

4.1 Finding $\boldsymbol{P}_1$

The position of joint 1 is represented by $\boldsymbol{P}_1$. The Denavit-Hartenberg parameters for link 1 has non-zero a and d parameters, which means that there is an offset from the rotational axis of joint 1, which is seen in Figure 2b. This point is on the point pair $\boldsymbol{Q}_1$ that is found by intersecting a sphere with two planes as follows:

$$\begin{aligned} \boldsymbol{S}_0^* &= e_0 - \frac{1}{2}\rho^2 e_\infty, \quad \rho^2 = d_1^2 + a_1^2 \\ \mathbf{\Pi}_{1x}^* &= e_3 + d_1 e_\infty \\ \boldsymbol{Q}_1 &= (\boldsymbol{S}_0^* \wedge \mathbf{\Pi}_{1x}^*) \cdot \mathbf{\Pi}_c \end{aligned} \tag{29}$$

This point pair consists of the two possible solutions for $\boldsymbol{P}_1$. One solution corresponds to robot facing towards the end-effector, while the other corresponds to the robot facing away from the end-effector. The solution for $\boldsymbol{P}_1$ is selected according to

$$\boldsymbol{P}_{1\pm} = \frac{\boldsymbol{Q}_1 \pm \sqrt{\boldsymbol{Q}_1^2}}{-e_\infty \cdot \boldsymbol{Q}_1} \tag{30}$$

$$\boldsymbol{P}_1 = \begin{cases} \boldsymbol{P}_{1+} & \text{if } k_{fb}(\boldsymbol{P}_{1+} \cdot \boldsymbol{P}_e) > k_{fb}(\boldsymbol{P}_{1-} \cdot \boldsymbol{P}_e) \\ \boldsymbol{P}_{1-} & \text{otherwise} \end{cases} \tag{31}$$

Figure 6 shows the geometric objects in Equation 29 and the selected $\boldsymbol{P}_1$.

4.2 Finding $\boldsymbol{P}_2$

$\boldsymbol{P}_2$ will be on the circle $\boldsymbol{C}_2$, which is the intersection of the two spheres

$$\begin{aligned} \boldsymbol{S}_1^* &= \boldsymbol{P}_1 - \frac{1}{2}a_2^2 e_\infty \\ \boldsymbol{S}_w^* &= \boldsymbol{P}_w - \frac{1}{2}(d_4^2 + a_3^2) e_\infty \end{aligned} \tag{32}$$

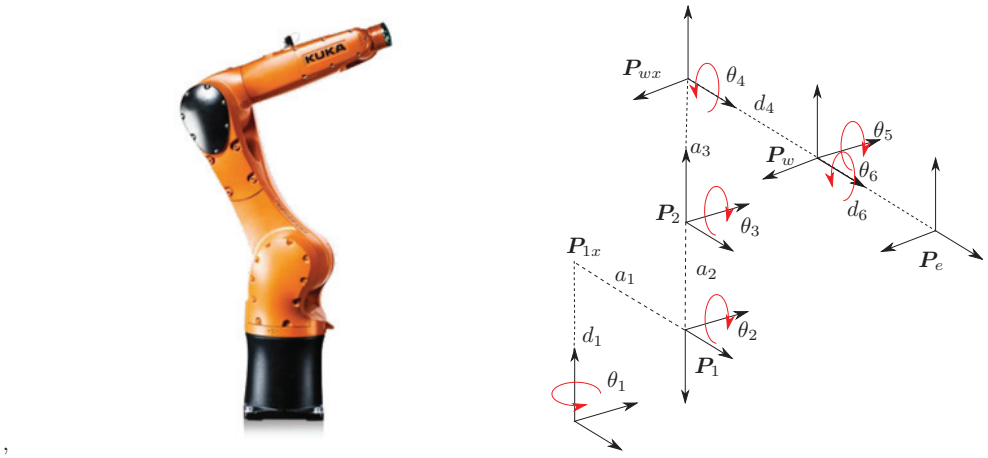
where $\boldsymbol{S}_1$ has center point $\boldsymbol{P}_1$, and $\boldsymbol{S}_w$ is centered in $\boldsymbol{P}_w$. Then the intersection of $\boldsymbol{C}_2$ with the vertical plane $\mathbf{\Pi}_c$ will give a point pair $\boldsymbol{Q}_2$, according to

$$\begin{aligned} \boldsymbol{C}_2^* &= \boldsymbol{S}_1^* \wedge \boldsymbol{S}_w^* \\ \boldsymbol{Q}_2 &= \boldsymbol{C}_2^* \cdot \mathbf{\Pi}_c \end{aligned} \tag{33}$$

This is shown in Figure 4. The points in $\boldsymbol{Q}_2$ are the two possible solutions for $\boldsymbol{P}_2$, and the solution is selected depending on the parameter elbow up or elbow down, and is given by

$$\boldsymbol{P}_2 = \frac{\boldsymbol{Q}_2 - k_{ud}\sqrt{\boldsymbol{Q}_2^2}}{-e_\infty \cdot \boldsymbol{Q}_2} \tag{34}$$

Both configurations are shown in Figure 4a and Figure 4b.

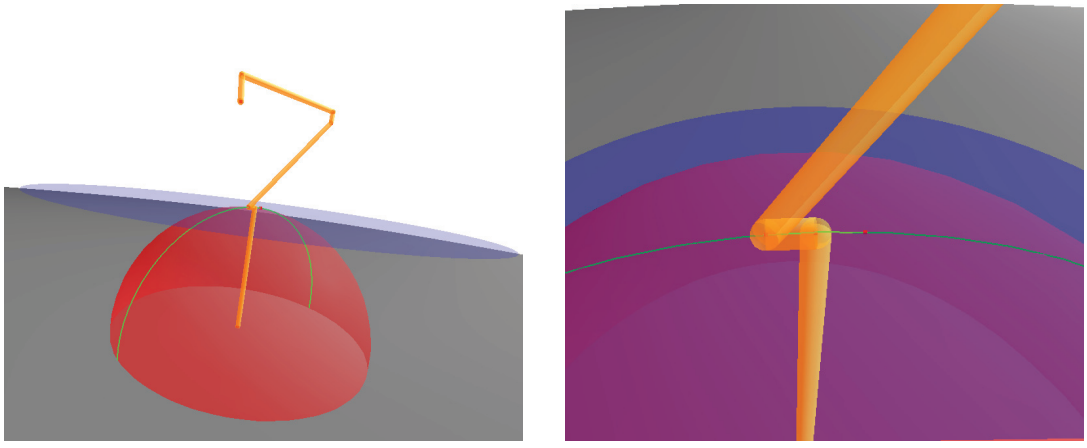


,

(a) The Agilus KR6 R900 sixx robot. Image taken from www.kuka-robotics.com

(b) The joint frames of the Agilus robot. The joint position is $\boldsymbol{q} = [0, -\frac{\pi}{2}, \frac{\pi}{2}, 0, 0, 0]^{\mathrm{T}}$.

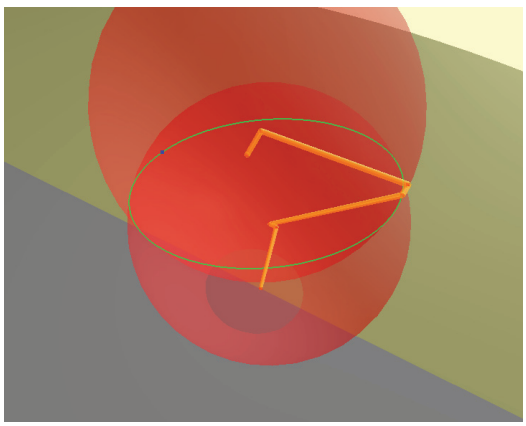
Figure 2: Overview of the Agilus KR6 R900 sixx robot.



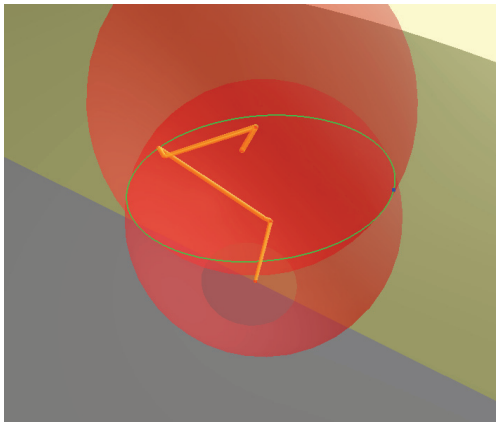
(a) A overview image of the robot and the geometric objects used to find $\boldsymbol{P}_1$

(b) A closer view of finding $\boldsymbol{P}_1$

Figure 3: The red sphere is $\boldsymbol{S}_0$, the blue plane is $\boldsymbol{\Pi}_{1x}$, the green circle is generated from $\boldsymbol{S}_0 \wedge \boldsymbol{\Pi}_c$, and the red point pair is $\boldsymbol{Q}_1$, where one is picked to be $\boldsymbol{P}_1$.



(a) The robot and the geometric objects used to find $\boldsymbol{P}_2$. The robot is configured with elbow up



(b) The robot and the geometric objects used to find $\boldsymbol{P}_2$. The robot is configured with elbow down

Figure 4: The red spheres are $\boldsymbol{S}_1$ and $\boldsymbol{S}_w$, the yellow plane is $\boldsymbol{\Pi}_{1x}$, the green circle is generated from $\boldsymbol{C}_2$, and the blue point pair is $\boldsymbol{Q}_2$, where one is picked to be $\boldsymbol{P}_2$.

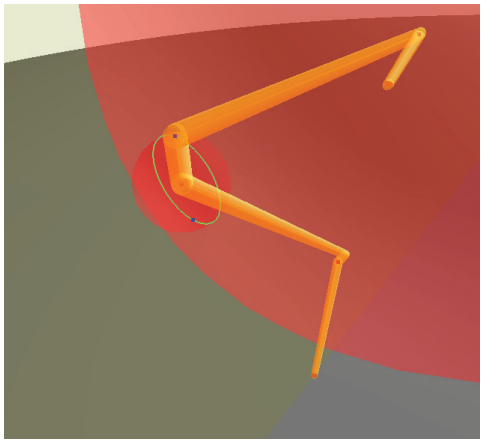


Figure 5: The robot and the geometric objects used to find $\boldsymbol{P}_{wx}$. The red spheres are $\boldsymbol{S}_2$ and $\boldsymbol{S}_w$, the yellow plane is $\boldsymbol{\Pi}_{1x}$, the green circle is generated from $\boldsymbol{C}_{wx}$, and the blue point pair is $\boldsymbol{Q}_{wx}$, where one is picked to be $\boldsymbol{P}_{wx}$.

4.3 Calculating the remaining kinematics

The Agilus has an offset a_3 from the joint position $\boldsymbol{P}_2$ to the offset point $\boldsymbol{P}_{wx}$, as shown in Table 1. The point $\boldsymbol{P}_{wx}$ is found using the horizon technique from Section 3.6, which gives

$$\begin{aligned} \boldsymbol{S}_2^* &= \boldsymbol{P}_2 - \frac{1}{2}a_3^2 e_\infty \\ \boldsymbol{K}_w^* &= \boldsymbol{P}_w - (\boldsymbol{P}_2 \cdot \boldsymbol{S}_2^*)e_\infty \\ \boldsymbol{C}_{wx}^* &= \boldsymbol{K}_w^* \wedge \boldsymbol{S}_w^* \\ \boldsymbol{Q}_{wx} &= \boldsymbol{C}_{wx}^* \cdot \boldsymbol{\Pi}_c \end{aligned} \tag{35}$$

Here the solutions for $\boldsymbol{P}_{wx}$ are the points of the point pair $\boldsymbol{Q}_{wx}$, and the solution is selected according to the arm geometry from the calculations

$$\boldsymbol{\Pi}_{2wc} = \boldsymbol{P}_2 \wedge \boldsymbol{P}_w \wedge \boldsymbol{\Pi}_c^* \wedge e_\infty$$

$$\boldsymbol{P}_{wx} = \begin{cases} \boldsymbol{P}_{wx+} & \text{if} \quad k_{fb}\boldsymbol{P}_{wx+} \cdot \boldsymbol{\Pi}_{2wc}^* > 0 \\ \boldsymbol{P}_{wx-} & \text{otherwise} \end{cases} \tag{36}$$

where

$$\boldsymbol{P}_{wx\pm} = \frac{\boldsymbol{Q}_{wx} \pm \sqrt{\boldsymbol{Q}_{wx}^2}}{-e_\infty \cdot \boldsymbol{Q}_{wx}} \tag{37}$$

Equation 35 and $\boldsymbol{P}_{wx}$ are shown in in Figure 5.

4.4 Finding the joint angles

The link configurations have now been determined from the end-effector configuration, and as remarked by [Dorst et al. (2009)], this is sufficient for graphical

rendering. The next step is to determine the joint angles. In previous works this has typically been done by calculating the cosine of the joint angle. This will not give consistent signs for the joint angles corresponding to the different solution for the arm. This problem is solved here, and it is shown how to determine the quadrant of the angle, and also to keep track of the different solutions.

To do this it is necessary to define the rotation plane of each joint, and the vectors defining the rotation of the joint. The point $\boldsymbol{P}_{1x} = C(d_1 e_3)$ and the following lines are defined:

$$\begin{aligned} \boldsymbol{L}_{1x1} &= \boldsymbol{P}_{1x} \wedge \boldsymbol{P}_1 \wedge e_\infty \\ \boldsymbol{L}_{12} &= \boldsymbol{P}_1 \wedge \boldsymbol{P}_2 \wedge e_\infty \\ \boldsymbol{L}_{wxw} &= \boldsymbol{P}_{wx} \wedge \boldsymbol{P}_w \wedge e_\infty \\ \boldsymbol{L}_{we} &= \boldsymbol{P}_w \wedge \boldsymbol{P}_e \wedge e_\infty \end{aligned} \tag{38}$$

The rotation plane of θ_1 is $\hat{\boldsymbol{N}}_{\theta_1} = e_1 \wedge e_2$, which is the horizontal base plane, while the rotation plane for θ_2 and θ_3 is found from the $\boldsymbol{\Pi}_c$ using Equation 21. Next, the rotation plane for θ_4 it is found from $\boldsymbol{L}_{wxw}$ using Equation 20, while for θ_5 the rotation plane is parallel to the plane $\boldsymbol{L}_{wxw} \wedge \boldsymbol{P}_e$, and its rotation depends on if it is flipped, i.e. k_{fn}. Finally, $-\boldsymbol{a}_e^+$ is the rotation plane for θ_6. The joint angles can then be found from Equation 17 using the parameters given in Table 3.

4.5 Singularities for the Agilus

There are two singularities in the given model, which correspond to the physical singularities of the robot.

In the wrist singularity, $\boldsymbol{P}_e$ will be on the line $\boldsymbol{L}_{wxw}$. Then the the rotation plane $\hat{\boldsymbol{N}}_{\theta_5}^{-1}$ becomes undefined since $\boldsymbol{L}_{wxw} \wedge \boldsymbol{P}_e = 0$.

In the shoulder singularity the point $\boldsymbol{P}_w$ will be on the vertical line defined by e_3, and the plane $\boldsymbol{\Pi}_c$ becomes undefined since $e_0 \wedge e_3 \wedge \boldsymbol{P}_w \wedge e_\infty = 0$.

5 Inverse Kinematics for the UR5

The input to the inverse kinematics of the UR5 robot is $\mathbf{p}_e$, $\mathbf{n}_e$, $\mathbf{s}_e$ and $\mathbf{a}_e$ as for the Agilus robot. The conformal representation of the end effector position and the position of joint 5 is found from

$$\begin{aligned} \boldsymbol{P}_e &= C(\boldsymbol{p}_e) \\ \boldsymbol{P}_5 &= C(\boldsymbol{p}_e - d_6 \boldsymbol{a}_e) \end{aligned} \tag{39}$$

The configuration parameters are defined as

$$k_{ud} = \begin{cases} 1 & \text{if elbow up} \\ -1 & \text{if elbow down} \end{cases} \tag{40}$$

$$k_{lr} = \begin{cases} 1 & \text{if shoulder right} \\ -1 & \text{if shoulder left} \end{cases} \tag{41}$$

$$k_{fn} = \begin{cases} 1 & \text{if wrist is not flipped} \\ -1 & \text{if wrist is flipped} \end{cases} \tag{42}$$

First the vertical plane $\boldsymbol{\Pi}_c$ through joints 1, 2, 3 and 4 is found. This is done by finding the point $\boldsymbol{P}_c$ with an offset d_4 from $\boldsymbol{P}_5$. The calculation is done with the horizon technique to find the circle $\boldsymbol{C}_{5k}$ according to

$$\begin{aligned} \boldsymbol{S}_c^* &= \boldsymbol{P}_5 - \frac{1}{2} d_4^2 e_\infty \\ \boldsymbol{K}_0^* &= e_0 - (\boldsymbol{S}_c^* \cdot e_0) e_\infty \\ \boldsymbol{C}_{5k}^* &= \boldsymbol{S}_c^* \wedge \boldsymbol{K}_0^* \end{aligned} \tag{43}$$

Then the point pair $\boldsymbol{Q}_c$ with the two solutions for $\boldsymbol{P}_c$ is found by intersection $\boldsymbol{C}_{5k}$ with the horizontal plane through $\boldsymbol{P}_5$:

$$\boldsymbol{Q}_c = \boldsymbol{C}_{5k}^* \cdot (\boldsymbol{P}_5 \wedge e_1 \wedge e_2 \wedge e_\infty) \tag{44}$$

The solution for $\boldsymbol{P}_c$ is selected depending on the parameter for shoulder right or shoulder left using

$$\boldsymbol{P}_c = \frac{\boldsymbol{Q}_c + k_{lr} \sqrt{\boldsymbol{Q}_c^2}}{-e_\infty \cdot \boldsymbol{Q}_c} \tag{45}$$

When the solution for $\boldsymbol{P}_c$ has been selected the vertical plane $\boldsymbol{\Pi}_c$ is found from

$$\boldsymbol{\Pi}_c = e_0 \wedge e_3 \wedge \boldsymbol{P}_c \wedge e_\infty \tag{46}$$

Figure 7 shows the geometric objects in Equation 43 and the point pair $\boldsymbol{Q}_c$.

5.1 Finding $\boldsymbol{P}_3$ and $\boldsymbol{P}_4$

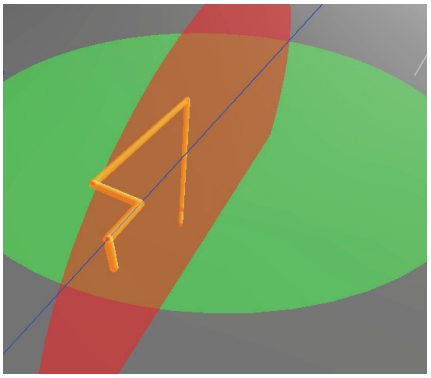
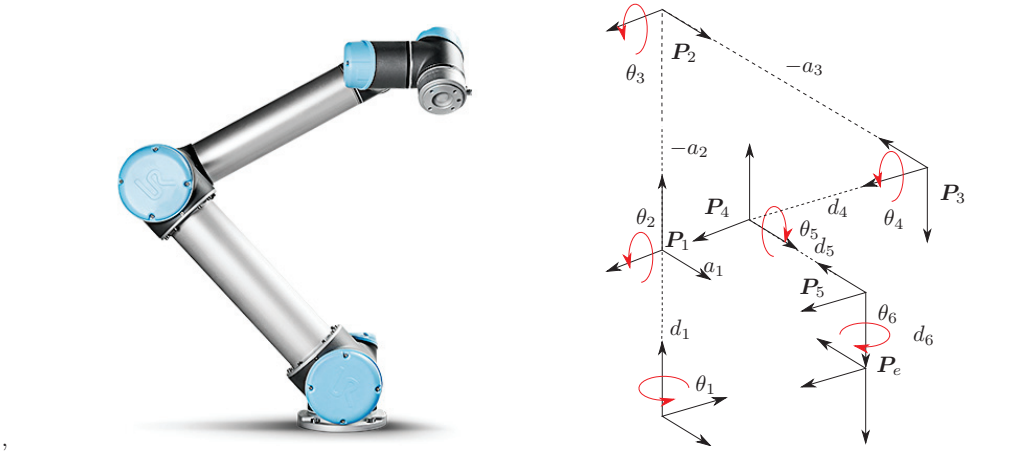


Figure 8: The green planes is $\boldsymbol{\Pi}_{c\perp}$ and the red plane is $\boldsymbol{\Pi}_{c\parallel}$ and the blue line is $\boldsymbol{L}_{45}$

θ_i	$\boldsymbol{a}_{\theta_i}$	$\boldsymbol{b}_{\theta_i}$	$\boldsymbol{N}_{\theta_i}$	offset
1	$k_{fb}\boldsymbol{\Pi}_c^*$	$-e_2$	$e_1 \wedge e_2$	0
2	$(\boldsymbol{L}_{1x1} \cdot e_0) \cdot e_\infty$	$(\boldsymbol{L}_{12} \cdot e_0) \cdot e_\infty$	$k_{fb}(\boldsymbol{\Pi}_c \cdot e_0) \cdot e_\infty$	0
3	$(\boldsymbol{L}_{12} \cdot e_0) \cdot e_\infty$	$(\boldsymbol{L}_{wxw} \cdot e_0) \cdot e_\infty$	$k_{fb}(\boldsymbol{\Pi}_c \cdot e_0) \cdot e_\infty$	$-\frac{\pi}{2}$
4	$-\boldsymbol{\Pi}_c^*$	$-k_{fb}k_{fn}\left((\boldsymbol{L}_{wxw} \wedge \boldsymbol{P}_e)^* \wedge e_0\right) \cdot e_\infty$	$(\boldsymbol{L}_{wxw}^* \wedge e_0) \cdot e_\infty$	0
5	$(\boldsymbol{L}_{we} \cdot e_0) \cdot e_\infty$	$(\boldsymbol{L}_{wxw} \cdot e_0) \cdot e_\infty$	$k_{fn}\left((\boldsymbol{L}_{wxw} \wedge \boldsymbol{P}_e) \cdot e_0\right) \cdot e_\infty$	0
6	$\left((\boldsymbol{L}_{wxw} \wedge \boldsymbol{P}_e)^* \wedge e_0\right) \cdot e_\infty$	$-\boldsymbol{s}_e$	$-\boldsymbol{a}_e^+$	0

Table 3: Joint angle parameters for the Agilus robot. It can be verified that the dual of $\hat{\boldsymbol{N}}_{\theta_i}$ is the rotational axis $\boldsymbol{z}_{i-1}$ of the Denavit-Hartenberg convention. Note that the table shows the non-normalized bivectors $\boldsymbol{N}_{\theta_i}$.

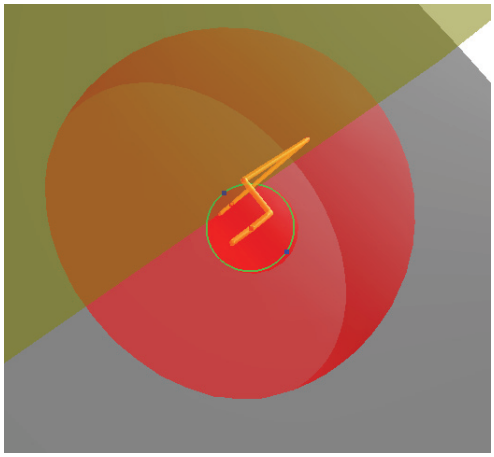


,

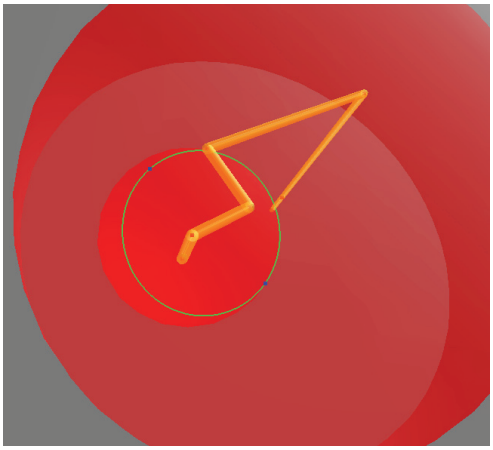
(a) The UR5 robot. Image taken from www.universal-robots.com

(b) The joint frames of the UR5 robot. The joint position is $\boldsymbol{q} = [0, -\frac{\pi}{2}, -\frac{\pi}{2}, -\frac{\pi}{2}, \frac{\pi}{2}, 0]^{\mathrm{T}}$

Figure 6: Overview of the UR5 robot.



(a) Overview of the UR5 robot, $\boldsymbol{Q}_c$ and $\boldsymbol{\Pi}_c$



(b) Closer view of the UR5 robot and $\boldsymbol{Q}_c$

Figure 7: The red spheres are $\boldsymbol{S}_c$ and $\boldsymbol{K}_0$, the green circle is $\boldsymbol{C}_{5k}$, $\boldsymbol{Q}_c$ is the blue point pair, and $\boldsymbol{\Pi}_c$ is the yellow plane.

The plane $\boldsymbol{\Pi}_{c\|}$ is defined as the plane that is parallel to $\boldsymbol{\Pi}_c$ and that contains the points $\boldsymbol{P}_4$ and $\boldsymbol{P}_5$. This plane is found in the dual form with a distance $\boldsymbol{P}_5 \cdot \boldsymbol{\Pi}_c$ from $\boldsymbol{\Pi}_c$ according to

$$\boldsymbol{\Pi}^*_{c\|} = \boldsymbol{\Pi}^*_c + (\boldsymbol{P}_5 \cdot \boldsymbol{\Pi}^*_c)e_\infty \tag{47}$$

The next step is to calculate the line through $\boldsymbol{P}_4$ and $\boldsymbol{P}_5$ from

$$\begin{aligned} \boldsymbol{\Pi}_{56\perp} &= (\boldsymbol{P}_5 \wedge \boldsymbol{P}_6)^* \wedge e_\infty \\ \hat{\boldsymbol{n}}_{56\perp} &= -\frac{(\boldsymbol{\Pi}_{56\perp} \cdot e_0) \cdot e_\infty}{\|(\boldsymbol{\Pi}_{56\perp} \cdot e_0) \cdot e_\infty\|} \\ \boldsymbol{\Pi}_{c\perp} &= \boldsymbol{P}_5 \wedge \hat{\boldsymbol{n}}_{56\perp} \wedge e_\infty \\ \boldsymbol{L}^*_{45} &= \boldsymbol{\Pi}^*_{c\|} \wedge \boldsymbol{\Pi}^*_{c\perp} \end{aligned} \tag{48}$$

where $\boldsymbol{\Pi}_{c\perp}$ is a plane containing $\boldsymbol{P}_4$ and $\boldsymbol{P}_5$ and which normal is perpendicular to the normal of $\boldsymbol{\Pi}_c$. It is noted that $\hat{\boldsymbol{n}}_{56\perp} = \boldsymbol{a}^+_e = \boldsymbol{s}_e \wedge \boldsymbol{n}_e$.

The solutions for $\boldsymbol{P}_4$ are then given by the point pair $\boldsymbol{Q}_4$, which is the intersection of the line $\boldsymbol{L}_{45}$ and the sphere $\boldsymbol{S}_5$ with center point in $\boldsymbol{P}_5$ and radius d_5. This is calculated from

$$\begin{aligned} \boldsymbol{S}^*_5 &= \boldsymbol{P}_5 - \frac{1}{2}d_5^2 e_\infty \\ \boldsymbol{Q}_4 &= \boldsymbol{L}_{45} \cdot \boldsymbol{S}^*_5 \\ \boldsymbol{P}_4 &= \frac{\boldsymbol{Q}_4 + k_{fn}\sqrt{\boldsymbol{Q}_4^2}}{-e_\infty \cdot \boldsymbol{Q}_4} \end{aligned} \tag{49}$$

Next, the solutions for $\boldsymbol{P}_3$ are given by the point pair $\boldsymbol{Q}_3$, which is the intersection of the line $\boldsymbol{L}_{34}$ and the sphere $\boldsymbol{S}_4$ with center point in $\boldsymbol{P}_4$ and radius d_4. This is calculated from

$$\begin{aligned} \boldsymbol{S}^*_4 &= \boldsymbol{P}_4 - \frac{1}{2}d_4^2 e_\infty \\ \boldsymbol{L}_{34} &= \boldsymbol{P}_5 \wedge \boldsymbol{\Pi}^*_c \wedge e_\infty \\ \boldsymbol{Q}_3 &= \boldsymbol{S}^*_4 \cdot \boldsymbol{L}_{34} \\ \boldsymbol{P}_3 &= \frac{\boldsymbol{Q}_3 - k_{lr}\sqrt{\boldsymbol{Q}_3^2}}{-e_\infty \cdot \boldsymbol{Q}_3} \end{aligned} \tag{50}$$

5.2 Finding $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$

$\boldsymbol{P}_1$ is computed from

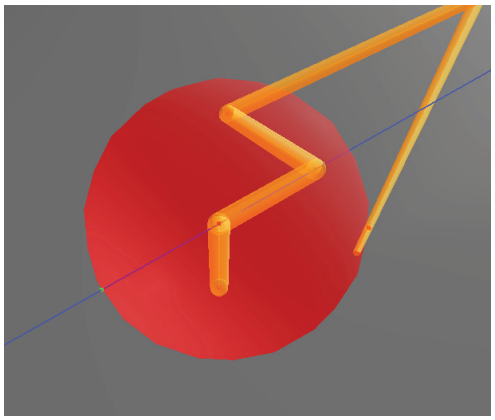
$$\boldsymbol{P}_1 = C(d_1 e_3) \tag{51}$$

The solutions for the point $\boldsymbol{P}_2$ are then given by the point pair $\boldsymbol{Q}_2$, which is found as the intersection of the two spheres $\boldsymbol{S}_1$ and $\boldsymbol{S}_3$ and the vertical plane $\boldsymbol{\Pi}_c$, which is calculated from

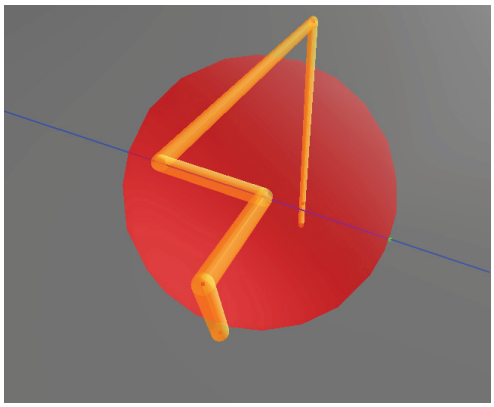
$$\begin{aligned} \boldsymbol{S}^*_1 &= \boldsymbol{P}_1 - \frac{1}{2}a_2^2 e_\infty \\ \boldsymbol{S}^*_3 &= \boldsymbol{P}_3 - \frac{1}{2}a_3^2 e_\infty \\ \boldsymbol{C}^*_2 &= \boldsymbol{S}^*_1 \wedge \boldsymbol{S}^*_3 \\ \boldsymbol{Q}_2 &= \boldsymbol{C}^*_2 \cdot \boldsymbol{\Pi}_c \end{aligned} \tag{52}$$

The solution is selected depending on the the parameter for elbow up or down according to

$$\boldsymbol{P}_2 = \frac{\boldsymbol{Q}_2 - k_{ud}\sqrt{\boldsymbol{Q}_2^2}}{-e_\infty \cdot \boldsymbol{Q}_2} \tag{53}$$



(a) The blue line is $\boldsymbol{L}_{45}$, the red sphere is $\boldsymbol{S}_5$ and the green point pair is $\boldsymbol{Q}_4$

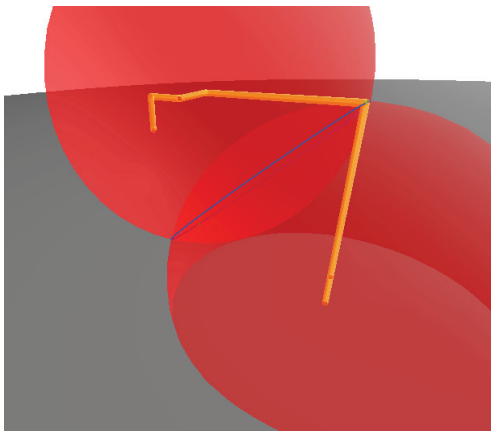


(b) The blue line is $\boldsymbol{L}_{34}$, the red sphere is $\boldsymbol{S}_4$ and the green point pair is $\boldsymbol{Q}_3$

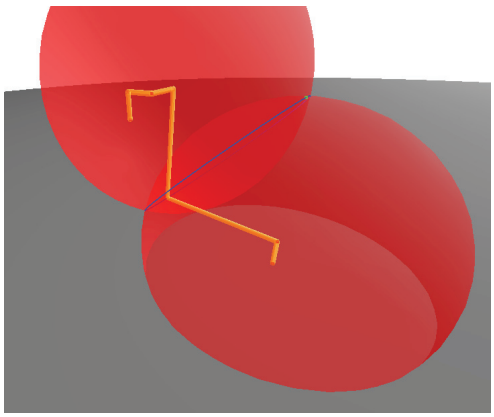
Figure 9: Figures showing the process of finding $\boldsymbol{P}_3$ and $\boldsymbol{P}_4$

θ_i	$\boldsymbol{a}_{\theta_i}$	$\boldsymbol{b}_{\theta_i}$	$\boldsymbol{N}_{\theta_i}$	offset
1	e_2	$-k_{lr}\boldsymbol{\Pi}_c^*$	$e_1 \wedge e_2$	0
2	$(\boldsymbol{L}_{01} \cdot e_0) \cdot e_\infty$	$(\boldsymbol{L}_{12} \cdot e_0) \cdot e_\infty$	$-k_{lr}(\boldsymbol{\Pi}_c \cdot e_0) \cdot e_\infty$	$-\frac{\pi}{2}$
3	$(\boldsymbol{L}_{12} \cdot e_0) \cdot e_\infty$	$(\boldsymbol{L}_{23} \cdot e_0) \cdot e_\infty$	$-k_{lr}(\boldsymbol{\Pi}_c \cdot e_0) \cdot e_\infty$	0
4	$(\boldsymbol{L}_{23} \cdot e_0) \cdot e_\infty$	$(\boldsymbol{L}_{45} \cdot e_0) \cdot e_\infty$	$-k_{lr}(\boldsymbol{\Pi}_c \cdot e_0) \cdot e_\infty$	$-\frac{\pi}{2}$
5	$k_{lr}\boldsymbol{\Pi}_c^*$	$-\boldsymbol{a}_e$	$(-\boldsymbol{L}_{45}^* \wedge e_0) \cdot e_\infty$	0
6	$(\boldsymbol{L}_{45} \cdot e_0) \cdot e_\infty$	$-\boldsymbol{s}_e$	$-\boldsymbol{a}_e^+$	0

Table 4: Joint angle parameters for the UR5 robot. It can be verified that the dual of $\hat{\boldsymbol{N}}_{\theta_i}$ is the rotational axis $\boldsymbol{z}_{i-1}$ of the Denavit-Hartenberg convention. Note that the table shows the non-normalized bivectors $\boldsymbol{N}_{\theta_i}$.



(a) The UR5 robot with the elbow up configuration



(b) The UR5 robot with the elbow down configuration

Figure 10: The two red spheres are $\boldsymbol{S}_1$ and $\boldsymbol{S}_3$, the blue circle is $\boldsymbol{C}_2$ and the green point pair is $\boldsymbol{Q}_2$

5.3 Finding the joint angles

Expression for calculating the configuration have now been established. The next step is to find expressions for the calculation of the joint angles using Equation 17.

The following lines are defined

$$\begin{aligned} \boldsymbol{L}_{01} &= e_0 \wedge e_3 \wedge e_\infty \\ \boldsymbol{L}_{12} &= \boldsymbol{P}_1 \wedge \boldsymbol{P}_2 \wedge e_\infty \\ \boldsymbol{L}_{23} &= \boldsymbol{P}_2 \wedge \boldsymbol{P}_3 \wedge e_\infty \end{aligned} \tag{54}$$

The rotation planes for θ_1 and θ_6 of the UR5 are the same as for the Agilus: $e_1 \wedge e_2$ and $-\boldsymbol{a}_e^+$ respectively.

The angles θ_2, θ_3 and θ_4 have the same rotation plane, which is parallel to $\boldsymbol{\Pi}_c$, while the angle θ_5 rotates around the line $\boldsymbol{L}_{45}$. Then Equation 21 and Equation 20 can be used, and the rotation planes are found as shown in Table 4.

Table 4 shows the parameters used in Equation 17 to calculate the joint angle for the UR5.

5.3.1 Singularities for the UR5

There are two singularity in this mathematical model, which are the same as the singularities of the robot.

The shoulder singularity occurs when $\boldsymbol{P}_c = \alpha e_3, \quad \forall \alpha$, which means that $\boldsymbol{P}_c$ is on the rotational axis of joint 1. Then $\boldsymbol{\Pi}_c$ in Equation 46 becomes undefined as $e_0 \wedge e_3 \wedge \boldsymbol{P}_c \wedge e_\infty = 0$.

The wrist singularity occurs when $\boldsymbol{\Pi}^*_{c\|} \wedge \boldsymbol{\Pi}^*_{c\perp} = 0$, which will be the case when $\theta_5 = \pm\frac{\pi}{2}$. Then the line $\boldsymbol{L}_{45}$ in Equation 48 becomes undefined.

6 Results

Analytic inverse kinematic solutions for the KUKA Agilus robot and the UR5 robot were implemented in the CluCalc software for calculation and display of geometric algebra. The files can be downloaded from https://github.com/ipk-ntnu/inverse_kinematics_using_cga. The solutions were extensively tested in simulations by interactively moving the robots over the whole workspace for different solutions of the type elbow up and down, shoulder left and right, and wrist flipped or not. The solutions were in particular tested close to the manipulator singularities.

The accuracy of the inverse kinematic solution was validated by calculating the homogeneous transformation matrix according to Equation 3 and comparing the result with the input parameters $\boldsymbol{n}_e$, $\boldsymbol{s}_e$, $\boldsymbol{a}_e$ and $\boldsymbol{p}_e$. The results were correct with accuracy close to machine precision over the whole workspace.

The programming of the solutions is focused on the intersection of geometric objects like lines, circles, planes and spheres that are readily displayed during programming, and this gave valuable intuitive support in the development of the calculations. Moreover, extensive testing over the workspace was facilitated by the 3D graphics.

7 Conclusion

Conformal geometric algebra has been used to develop analytical inverse kinematic solutions for the KUKA Agilus robot and the UR5 robot. The inverse kinematic solutions gave consistent signs for the angles for the different solutions of the robots. Compared to earlier work in conformal geometric algebra the proposed method handles link offsets and gives correct joint angles over the whole workspace for the different solutions related to shoulder left and right, elbow up and down and wrist flipped or not. The software solution can be ported to standard software like C or C++ for implementation in robot controllers. The method is fairly intuitive and easy to program once the machinery of conformal geometric algebra is mastered, and it provides a powerful tool for developing solutions for new robot geometries and other mechanisms like cranes and automatic topside drilling equipment.

References

Dorst, L., Fontijne, D., and Mann, S. *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2009. URL http://www.geometricalgebra.net/.

Hildenbrand, D. *Foundations of Geometric Algebra Computing*, volume 8 of *Geometry and Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. doi:10.1007/978-3-642-31794-1.

Hildenbrand, D., Bayro-Corrochano, E., and Zamora, J. Advanced geometric approach for graphics and visual guided robot object manipulation. *Proceedings - IEEE International Conference on Robotics and Automation*, 2005. 2005:4727–4732. doi:10.1109/ROBOT.2005.1570850.

Hildenbrand, D., Fontijne, D., Wang, Y., Alexa, M., and Dorst, L. Competitive runtime performance for inverse kinematics algorithms using conformal geometric algebra. *Eurographics conference*, 2006. pages 2006–2006. URL http://www.gaalop.de/dhilden{_}data/EG06{_}Performance.pdf.

Perwass, C. *Geometric Algebra with Applications in Engineering*, volume 4 of *Geometry and Computing*.

Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi:10.1007/978-3-540-89068-3.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. *Robotics: modelling, planning and control*. 2009. URL http://www.springer.com/fr/book/9781846286414.

Spong, M. W., Hutchinson, S., and M., V. *Robot Modeling and Control*. 2006. doi:10.1109/TAC.2006.890316.

Zamora, J. and Bayro-Corrochano, E. Inverse kinematics, fixation and grasping using conformal geometric algebra. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 2004. 4(1 1):3841–3846. doi:10.1109/IROS.2004.1390013.

9.2 Paper 2: Object Detection in Point Clouds using Conformal Geometric Algebra

By Aksel Sveier, Adam Leon Kleppe and Olav Egeland

This paper presents object detection using conformal geometric algebra. The paper adapts the RANSAC method to conformal geometric algebra, and shows how it performs in order to find geometric models using the RANSAC method.

The experiments show that it is possible to use this method to perform a separation so that the objects on a table can be separated. This is one of the steps that is performed in [32] and [33] in order to find a point cloud from a CAD model and compare it to a point cloud taken with a 3D camera.

9.3 Paper 3: Automated Assembly using 2D and 3D Cameras

By Adam Leon Kleppe, Asgeir Bjørkedal, Kristoffer Larsen and Olav Egeland

This paper presents some of the findings in the master thesis by Asgeir Bjørkedal and Kristoffer Larsen.

The experiments show that performing a real industrial application where sub-millimeter accuracy is required, the Kinect camera together with state-of-the-art descriptor methods were not accurate enough. Some earlier research and experiments also showed that the 2D camera was only suitable when given very specific settings and environments.

Bjørkedal and Larsen then tried to combine the two methods in order to achieve an accuracy that could not be achieved individually. The successful experiments showed that industrial applications could be solved with this technology.

robotics

MDPI

Article

Automated Assembly Using 3D and 2D Cameras

Adam Leon Kleppe *, Asgeir Bjørkedal †, Kristoffer Larsen † and Olav Egeland

Department of Mechanical and Industrial Engineering, Norwegian University of Science and Technology, N-7491 Trondheim, Norway; asgeir.92@gmail.com (A.B.); kristofferlars1@gmail.com (K.L.); olav.egeland@ntnu.no (O.K.)

* Correspondence: adam.l.kleppe@ntnu.no; Tel.: +47-9189-6882

† These authors contributed equally to this work.

Received: 31 March 2017; Accepted: 19 June 2017; Published: 27 June 2017

Abstract: 2D and 3D computer vision systems are frequently being used in automated production to detect and determine the position of objects. Accuracy is important in the production industry, and computer vision systems require structured environments to function optimally. For 2D vision systems, a change in surfaces, lighting and viewpoint angles can reduce the accuracy of a method, maybe even to a degree that it will be erroneous, while for 3D vision systems, the accuracy mainly depends on the 3D laser sensors. Commercially available 3D cameras lack the precision found in high-grade 3D laser scanners, and are therefore not suited for accurate measurements in industrial use. In this paper, we show that it is possible to identify and locate objects using a combination of 2D and 3D cameras. A rough estimate of the object pose is first found using a commercially available 3D camera. Then, a robotic arm with an eye-in-hand 2D camera is used to determine the pose accurately. We show that this increases the accuracy to < 1 mm and $< 1^\circ$. This was demonstrated in a real industrial assembly task where high accuracy is required.

Keywords: robotics; assembly; 3D vision; 2D vision

1. Introduction

Computer vision is frequently used in industry to increase the flexibility of automated production lines without reducing the efficiency and high accuracy that automated production requires.

Assembly applications benefit from computer vision in many ways. Production lines with frequent changeovers, which occurs in some industries, can benefit from computer vision to determine position and orientation of the parts on the production line, without the need of additional equipment [1]. Assembly production lines are mostly a very controlled and structured environment, which is suitable for computer vision methods, to make them perform more predictably [2].

Shadows and reflections are frequent problems in 2D computer vision [3], since the methods will yield different results if an object is viewed from different angles or if its orientation changes. The key to gaining accurate and predictable results is to have a good initial position of the camera relative to the object. 2D eye-in-hand cameras [4] can actively change the viewpoint of a camera to a scene. This makes it possible to view an object from a particular angle, no matter which orientation it has. By using an eye-in-hand camera in this way, more predictable results can be achieved than with a stationary camera. However, in order to do this, the camera must be moved to suitable position relative to the object, which has to be found first.

The rising use of 3D cameras gives the opportunity for different methods and approaches [5], mostly due to the availability of depth information, which makes it easier to determine properties such as shapes and occlusion, compared to a traditional 2D camera. However, commercially available 3D cameras lack precision, which can only be found in high-grade 3D laser scanners [6], leading to inaccurate measurements that are not sufficiently accurate for automated production. By using the 3D

camera to detect objects and determine a rough estimate of their position, the eye-in-hand camera can use these estimates to move to a suited initial position.

Within computer vision, there are several approaches on how to detect, classify and estimate poses of objects, both within 2D and 3D computer vision. Examples of these are voting-based algorithms, such as [7], human robot collaboration, such as [8], and probabilistic methods, such as [9]. However, these approaches focus more on successful recognition and computation time rather than accuracy of the pose estimation. This makes them ideal for pick and place algorithms such as [10], but not for accurate assembly tasks.

In this paper, we combine existing solutions from both 2D eye-in-hand and 3D computer vision in order to detect objects more predictably and with higher accuracy than either 3D or 2D methods separately. This system uses the Computer Aided Design (CAD) model of each object to render 3D views [11] and use these to determine the position and orientation of each object with sufficient accuracy to be able to assemble the objects using a robotic arm.

This paper is organized as follows. First, a brief presentation is given of some common computer vision methods that are used in the paper, followed by a description of the system that uses both 2D and 3D computer vision methods. Finally, the paper provides experimental results of an assembly task using one 3D camera, one eye-in-hand 2D camera and two robotic arms.

2. Preliminaries

In the paper, a point cloud $\boldsymbol{P}$ is a set of points $\boldsymbol{p}_i \in \mathbb{R}^3$, represented by Euclidean vectors, so that

$$\boldsymbol{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}. \tag{1}$$

RANSAC [12] is short for Random Sample Consensus and is an iterative method for estimating model parameters from a data set containing several outliers. Figure 1a shows a scene consisting of three objects placed on a table. The table is detected in Figure 1b using RANSAC with a plane estimation. The inliers are marked in light gray. Figure 1c shows the scene when the inliers have been removed.

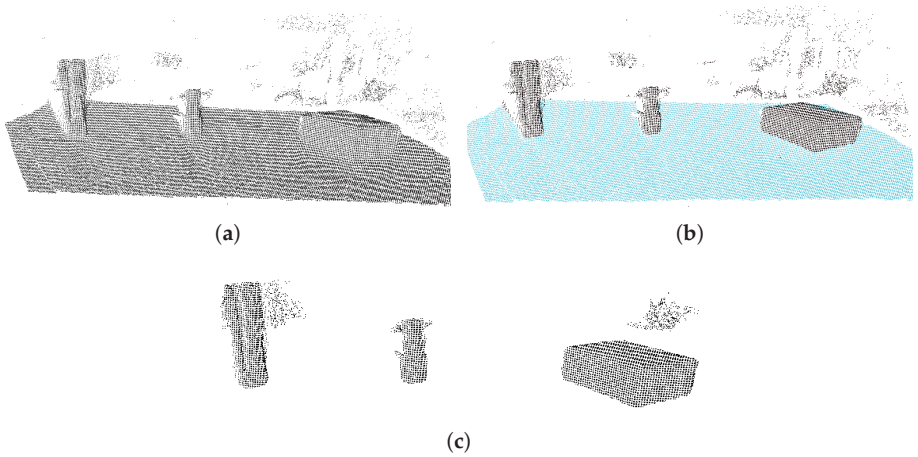


Figure 1. The RANSAC method performed on a table with objects. (**a**) A scene with multiple objects placed on a table; (**b**) the table surface is detected using RANSAC. The cyan points are inliers of the plane estimate; (**c**) The scene after removing the inliers, resulting in only the points representing the three objects.

The Iterative Closest Point (ICP) algorithm [13,14] is an iterative method for aligning two sets of point clouds. This is done by minimizing the distance between corresponding points.

One method that is suitable for calculating the rotation and translation is by using Singular Value Decomposition (SVD) to minimize the least squares error [15].

The scale invariant feature transform (SIFT) [16] is a method for matching image features. The method searches through images for interest points, which are points in an image surrounded by areas with sufficient information that makes it possible to distinguish them. The algorithm computes a SIFT descriptor for the image interest points, which is a histogram that can be used for matching.

3. Approach

The approach in this paper takes in a set of 3D CAD models and searches for these models within a scene. The system is divided into a 3D object detection system and a 2D object alignment system, as seen in Figure 2. The 3D detection system takes a 3D image and compares it to the given CAD models. The system identifies each object and calculates a rough estimate of the position and orientation of them. The estimates are then used as input to a 2D alignment system, where a 2D camera is mounted on a robotic arm, and uses 2D computer vision approaches to get a fine estimate of the positions and orientations of each object.

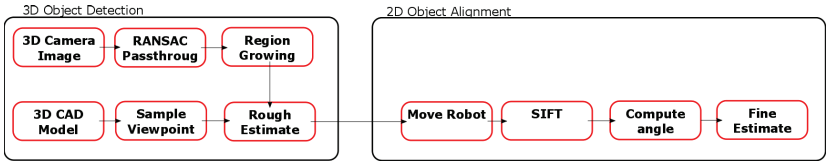


Figure 2. Overview of the flow of the system. It can be seen that the 3D Detection System calculates a rough position estimate given a set of CAD models, and this is fed into the 2D Alignment System, resulting in a fine position and orientation estimate.

3.1. 3D Object Detection

3.1.1. Viewpoint Sampling

A point cloud of a CAD model includes points on all sides of the 3D object, while a point cloud that forms a 3D camera will only have points on the part of the object, which is seen from the 3D camera.

To compare a point cloud captured by a 3D camera and a CAD model, it is important to make a comparison to the CAD model when viewed from the view point of the 3D camera. To do this, virtual 3D images of the object is generated from a CAD model from a selection of different viewpoints. Figure 3 shows a sample of the generated point clouds for an automotive part seen from seven different viewpoints.

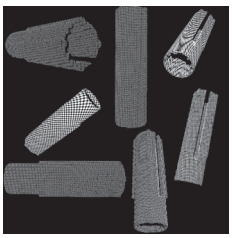


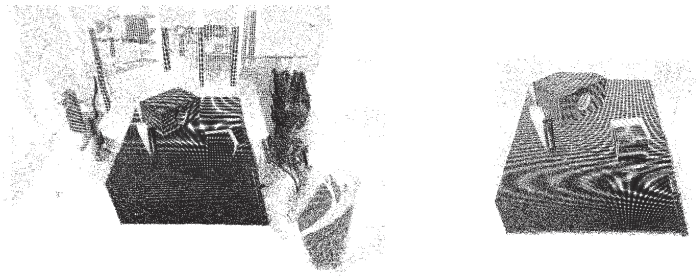
Figure 3. Point clouds of the same object seen from different viewpoints generated from a CAD model of the object.

Viewpoints are calculated using a tessellated sphere that is fit around the CAD model. Then, one viewpoint is rendered for each vertex of the tessellated sphere, and a point cloud is generated from each viewpoint.

For each generated point cloud, the local and global descriptors were calculated using the Fast Point Feature Histogram (FSFH) [17] approach and the Viewpoint Feature Histogram (VFH) [11] approach, respectively. The point clouds and their descriptors are then stored and labelled.

3.1.2. Removing Unqualified Points

The 3D camera will capture the whole scene. The resulting 3D image will include the objects, and in addition, points representing the foreground, background and the table where the objects are placed. In order to minimize the search area for the algorithms, the points that do not represent the object should be removed. The first step of removing unqualified points is to remove points outside a specified range. Since the objects are placed on a table, all points that are outside of the bounds of the table can safely be removed, see Figure 4.



(**a**) (**b**)

Figure 4. Before and after pictures of removing foreground and background points. (**a**) Raw point cloud captured by the 3D camera; (**b**) Image after removing unwanted points, which are the points outside the bounds of the table.

When these points are removed, the majority of the points in the point cloud will represent the table. The points representing the objects are on top of the table. This means that, by estimating the table plane, all points above this plane will represent the objects, while the points on the plane or under will be unqualified points. Since most of the point cloud has points representing the table, RANSAC can be used for estimating this plane.

A plane is determined by three points $\boldsymbol{p}_A$, $\boldsymbol{p}_B$ and $\boldsymbol{p}_C$ in the plane. Then, the normal of the plane, $\boldsymbol{v}$ is

$$(\boldsymbol{p}_B - \boldsymbol{p}_A) \times (\boldsymbol{p}_C - \boldsymbol{p}_A) = \boldsymbol{v} \tag{2}$$

and the distance from the origin is

$$d = \boldsymbol{v}^T \boldsymbol{p}_A. \tag{3}$$

A point $\boldsymbol{p}$ will have the distance $\delta = \frac{v^T p}{|v|}$ to the plane, where $\delta > 0$ when the point is in the direction of the normal of the plane.

To generate the model candidate for a plane in each iteration of RANSAC, three random points from the point cloud are picked: $\boldsymbol{p}_A$, $\boldsymbol{p}_B$ and $\boldsymbol{p}_C$. The candidate is then compared to the point cloud data to determine which points in the point cloud are inliers and which are outliers.

A point can be considered an inlier to the estimated plane if

$$\frac{|\boldsymbol{v}^T \boldsymbol{p}|}{|\boldsymbol{v}|} \leq \Delta, \tag{4}$$

where $\boldsymbol{p}$ is the given point, and Δ is a user-specified distance threshold. This threshold is the maximum distance away from the plane, where a point can be considered an inlier. All points outside of this threshold are considered outliers.

When the RANSAC algorithm is finished, the optimal plane estimate is found, and the inlier points of the plane are determined. With this information, the inlier points of the plane can be removed from the 3D image, as these represent the table and not the objects on top. In addition, the points below the table, which are characterized by $\boldsymbol{p}^T \boldsymbol{v} \leq \Delta$, can also be removed.

3.1.3. Object Detection

The remaining points in the point cloud will be part of an object or noisy outliers. In order to detect the objects, a region growing algorithm is used.

The region growing method finds points that are in close proximity and group them together. This is possible because the distance between two points on different objects are large relative to the distance between two points on the same object.

This algorithm results in one large group for each object in addition to several smaller groups containing noisy points. These smaller groups can be eliminated, based on their small size.

One new point cloud is then created for each of the remaining large groups. Each of these point clouds are the representation of an object on the table.

3.1.4. Object Alignment

From the previous step, there will be a number of point clouds, each representing an object on the table. It is not known which point cloud corresponds to each object, nor their position or orientation.

The next step is to find the viewpoint of the CAD models that best matches the point cloud for a particular object, which will give a rough estimate of position and orientation as well as the most likely identity of the object.

This is done by generating Fast Point Feature Histogram (FPFH) and Viewpoint Feature Histograms (VFH) descriptors for the point clouds of each object, and comparing these to viewpoint point clouds generated from the CAD models using the Sample Consensus-Initial Alignment (SAC-IA) [17] method.

SAC-IA uses the FPFH and VPF of each point cloud and their corresponding viewpoint point cloud to get the initial alignment of the object. SAC-IA is short for Sample Consensus Initial Alignment and is a variant of ICP that searches on a global scale rather than on a local scale as ICP does. This results in finding the viewpoint point cloud that best matches each object as well as the alignment between the viewpoint point cloud and the object point cloud.

The alignment from the SAC-IA method is a rough estimate, so the final step is to use ICP on the point clouds, resulting in an estimate of the alignment of each object in respect to the camera.

3.2. 2D Object Alignment

Due to the hardware limitations of the 3D-camera, the alignment estimates does not satisfy the requirements for assembly, so further estimations are required.

By placing a 2D-camera at the end-effector of a robotic arm (see Figure 5), the camera can be moved over each object and fine-tune their position and orientation. The robotic arm is positioned to a point right over the estimate calculated from the 3D computer vision system. This makes the camera be located approximately above the center of the object, and the camera can view the top of the object.

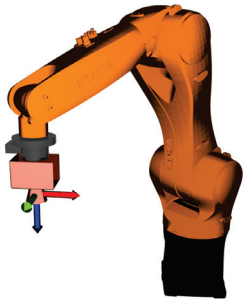


Figure 5. Overview of the 2D camera setup. It can be seen that the camera is placed at the end-effector of the robot, and it is pointing downward.

A 2D reference image is provided for each object, showing the top of the object, see Figure 6. When the 2D camera captures an image, it can be compared to the reference image using the SIFT method. The homography between the matched points is found, which makes it possible to find the transform between the reference image and the captured image. The reference image depicts the object in the center and at 0°, which means that the rotation of the homography is the orientation of the object, while the translation is the fine estimate of the position. This calculation can be run several times to converge to a better result, or to verify the current estimate:

$$v = p1 - p0, \tag{5}$$

$$\theta = \text{atan2}(v_y, v_x). \tag{6}$$

A sample from one of the experiments can be seen in Figure 7.

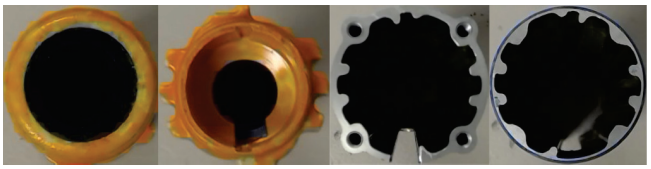


Figure 6. Reference images for each object. From left to right: The top of object A, the bottom of object A, the top of object B, the bottom of object B.

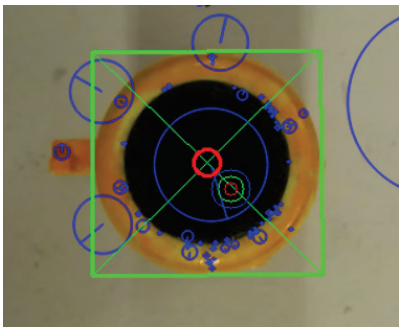


Figure 7. The green rectangle is the position of the reference image found in the captured image. The green circle is the current rough estimate from the 3D object detection system, while the red circle is the fine estimate of the position. The blue circles are the descriptors found with the SIFT method.

4. Experiments and Results

4.1. Setup

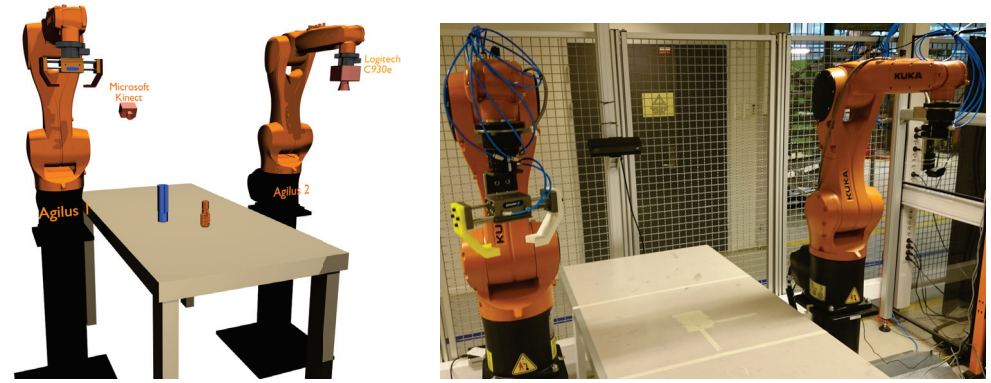
Tests of the assembly operation using the presented object detection procedures were performed in a robotic laboratory. The robotic cell was equipped with the following hardware:

- Two KUKA KR 6 R900 sixx (KR AGILUS) six-axis robotic manipulators (Augsburg, Germany).
- Microsoft Kinect™ One 3D depth sensor (Redmond, WA, USA).
- Logitech C930e web camera (Lausanne, Switzerland).
- Schunck PSH 22-1 linear pneumatic gripper (Lauffen, Germany).

Software used:

- Ubuntu 14.04 (Canonical, London, United Kindom).
- Point Cloud Library 1.7 (Willow Garage, Menlo Park, CA, USA).
- OpenCV 3.1 (Intel Corpiration, Santa Clara, CA, USA).
- Robot Operating System (ROS) Indigo (Willow Garage, Menlo Park, CA, USA).

The setup is shown in Figure 8. The pneumatic gripper was mounted at one of the robotic manipulators, while the web camera was mounted at the second manipulator. The Kinect One 3D camera was mounted behind the table and tilted towards the table top so that it could view the parts placed on the table. The position of the camera was calibrated in reference to the world frame of the robotic cell.



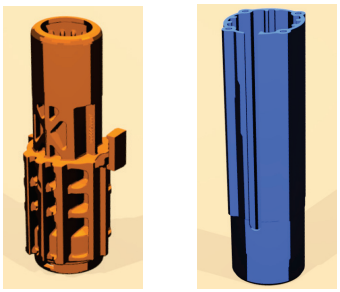
(**a**)Rendered representation of robotic cell used in the experiment.

(**b**)Physical cell used in the experiment.

Figure 8. Overview of the robotic cell, where the experiments were conducted. Here, there are two KUKA Agilus robots next to a table. The gripper can be viewed on the robot on the left, while the camera is on the right. Behind the table is the Microsoft Kinect One camera. (**a**) shows the rendered representation of the cell, while (**b**) shows the physical cell.

The assembly of two automotive parts was investigated in the experiment. These parts are shown in Figure 9.

A total of three different experiments were conducted to study the performance of a two-step alignment with initial 3D alignment and final 2D alignment. The first experiment was performed to determine the accuracy of the Kinect One 3D camera for the initial alignment, and the second experiment was to determine the accuracy of the 2D camera that was used in the final alignment. The last experiment was a full assembly of the two test objects using both 3D and 2D vision.



(**a**) (**b**)

Figure 9. The two parts used in all of the experiments. (**a**) Part A used in the experiment, rendered representation; (**b**) Part B used in the experiment, rendered representation.

4.2. Experiment 1: 3D Accuracy

The first experiment was designed to determine the accuracy of the 3D object detection system. The two objects of interest were positioned on the table in known locations. A grid of 5 cm $\times$ 5 cm squares was used to manually determine position the objects, as shown in Figure 10. The experiment was conducted 10 times on 16 different positions, and the resulting position from the 3D detection system was compared to the actual position. This was done with both of the objects.

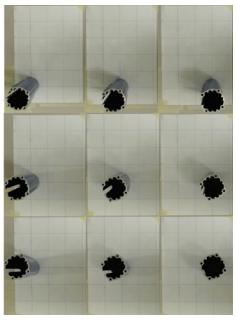


Figure 10. Top view of the grid and the positioning of the object. Here, nine arbitrary positions of the object is seen. For each of these positions, the rough estimate of the 3D object detection system is calculated.

4.2.1. Results

The experiment described above allowed the accuracy of the 3D detection procedure to be evaluated.

The test results show that the following positional deviations from the actual objects are shown in Tables 1–4.

Table 1. Minimum and maximum deviation between the true position and the estimated position of Object A. The results are based on 25 different estimates.

Min/Max Recorded Values	
Max $\triangle X$ [cm]	1.46
Max $\triangle Y$ [cm]	1.56
Min $\triangle X$ [cm]	0.43
Min $\triangle Y$ [cm]	0.08

Table 2. Accuracy of detecting Object A (measured in cm). The table shows the true position of Object A, the resulting estimate from the 3D object detection system, and the difference between the two.

Actual		**Measured**		**Absolute**	
X	Y	X	Y	$\lvert\triangle X\rvert$	$\lvert\triangle Y\rvert$
−5	5	−6.18	5.4	1.18	0.4
−5	10	−6.25	10.55	1.25	0.55
−5	15	−6.28	16.11	1.28	1.11
−5	20	−5.17	21.56	1.28	1.56
−10	5	−11.12	5.08	1.12	0.08
−10	10	−10.83	10.43	0.83	0.43
−10	15	−10.98	15.92	0.98	0.92
−10	20	−11.46	20.85	1.46	0.85
−15	5	−15.89	5.2	0.89	0.2
−15	10	−15.81	10.56	0.81	0.56
−15	15	−15.97	15.77	0.97	0.77
−15	20	−16.18	21.01	1.18	1.01
−20	5	−20.43	5.4	0.43	0.4
−20	10	−20.68	10.72	0.68	0.72
−20	15	−20.72	16.27	0.72	1.27
−20	20	−21.18	21.38	1.18	1.38

Table 3. Minimum and maximum deviation between the true position and the estimated position of Object B. The results are based on 25 different estimates.

Min/Max Recorded Values	
Max $\triangle X$ [cm]	1.43
Max $\triangle Y$ [cm]	1.96
Min $\triangle X$ [cm]	0.1
Min $\triangle Y$ [cm]	0.06

Table 4. Accuracy of detecting Object B (measured in cm). The table shows the true position of Object B, the resulting estimate from the 3D object detection system, and the difference between the two.

Actual		**Measured**		**Absolute**	
X	Y	X	Y	$\lvert\triangle X\rvert$	$\lvert\triangle Y\rvert$
−5	5	−5.76	5.16	0.76	0.16
−5	10	−6.12	10.8	1.12	0.8
−5	15	−5.98	15.94	0.98	0.94
−5	20	−6.17	20.88	1.17	0.88
−10	5	−10.65	5.47	0.65	0.47
−10	10	−10.62	10.21	0.62	0.21
−10	15	−10.73	15.81	0.73	0.81
−10	20	−10.91	20.79	0.91	0.79
−15	5	−15.22	5.46	0.22	0.46
−15	10	−15.46	10.62	0.46	0.62
−15	15	−15.71	16.2	0.71	1.2
−15	20	−15.85	21.14	0.85	1.14
−20	5	−20.1	5.43	0.1	0.43
−20	10	−20.73	10.06	0.73	0.06
−20	15	−20.26	16.35	0.26	1.35
−20	20	−21.43	21.96	1.43	1.96

The test results for the initial alignment show that the maximum positional error from the 3D measurements for both the x- and y-axis is below 2 cm. This is acceptable as a first step to make it possible to perform a final 2D alignment to achieve the required industrial accuracy.

4.3. Experiment 2: 2D Stability

The accuracy of a 2D computer vision method is related to the stability of the object detection, and this is directly related to the amount of good and repeatable keypoints detected in the reference and captured 2D image. If the detected keypoints differ every time an image is captured, the homography matrix computed from the feature correspondences will influence the computation of the object orientation significantly.

In order to ensure that this will not be a restricting factor in the assembly operation, an experiment was performed. The experiment was performed by positioning the object of interest at the table with two given orientations, 0° and 90°. The manipulator with the 2D camera in an eye-in-hand arrangement was moved to a distance from the object along the z-axis empirically chosen based on the rate of successful matching using SIFT. The detected object center is then aligned with the camera optical center. For every object, the angle of orientation was calculated based on the results of the 2D vision methods. This was done every time the camera captured an image. The mean was calculated for 10 measurements until the data set consists of 25 data points. The difference in degrees between minimum and maximum orientation estimates were used to determine the accuracy of the system.

The stability was first tested using SIFT, and it was then compared to using a hybrid algorithm, where SIFT keypoints were used with the Speeded Up Robust Feature (SURF) descriptor [18].

4.3.1. Results

The experiment described above yields the results shown in Tables 5 and 6.

Table 5. The difference between the maximum and minimum measured orientations for Object A. The first table is the deviation between the maximum and minimum angle when the object is positioned at 0°, both with using SIFT and with a SIFT/SURF hybrid. The second table is when the object is positioned at 90°. The measurements are given in degrees.

0 Degrees		**−90 Degrees**	
SIFT	SIFT/SURF	SIFT	SIFT/SURF
1.7469	5.4994	1.1102	7.9095

Table 6. The difference between the maximum and minimum measured orientations for Object B. The first table is the deviation between the maximum and minimum angle when the object is positioned at 0°, both with using SIFT, and with a SIFT/SURF hybrid. The second table is when the object is positioned at 90°. The measurements are given in degrees.

0 Ddegree		**−90 Degrees**	
SIFT	SURF	SIFT	SURF
0.07888	0.2041	0.1721	0.1379

It is evident from these results that the orientation of object A is the hardest to detect with certainty. Detection of object B is much more stable. This also shows that, using the SIFT method, one can acquire an accuracy of $< 2°$. The mean error was $< 1°$, which is acceptable for assembly.

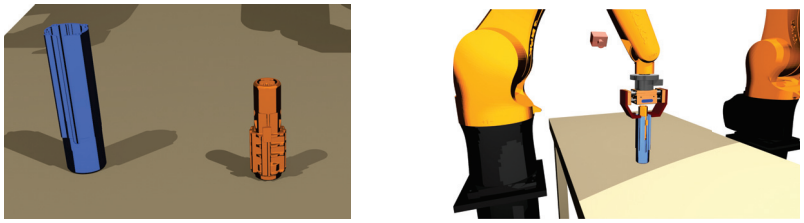
4.4. Experiment 3: Full Assembly

Based on the results from the previous experiments, a full assembly operation was be performed in an experiment. The procedure if the experiment was as follows:

1. Place the two objects to be assembled at random positions and orientations on the table.
2. Run the initial 3D alignment described in described in Section 3.

3. Perform the final 2D alignment by moving the robot in position above the part found in the initial alignment.
4. Move the robotic manipulator with the gripper to the estimated position of the first part, and pick it up. The manipulator then moved the part to the estimated pose of the second part to assemble the two parts.

These three steps were repeated for 10 unique assembly operations. The assembly operations are only considered as a success if the parts could be assembled without the use of force. An overview is shown in Figure 11.



(**a**) (**b**)

Figure 11. Overview of the assembly operation. (**a**) A rendered image of the initial position of the objects; (**b**) A rendered image of the final position of the objects. It can be seen that the orange object should be assembled inside the blue object.

4.4.1. Results

The assembly experiment is performed for 10 unique positions and orientations of object A and object B as described in Section 4.4. One of the 3D object detection results are visualized in Figure 12, while one of the results from the 2D object alignment is shown in Figure 13.

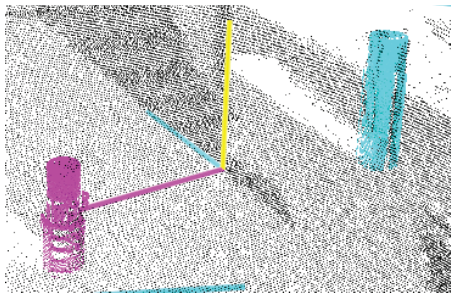


Figure 12. Results from the 3D object detection method. The method successfully classifies each object, and determines a rough estimate of their position.

Correction of the object position is performed using the 2D object detection and aligns the object center with the camera optical center as illustrated in Figure 14. The robotic end-effector pose is retrieved in world coordinates and the orientation is calculated.

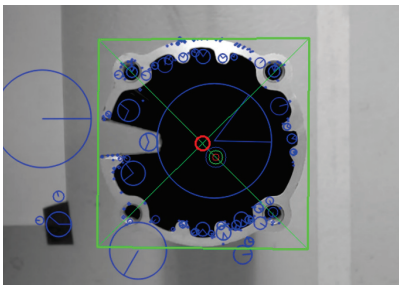


Figure 13. The small, green circle is the position determined by the 3D object detection. Using this estimate, the method can successfully detect a fine-tuned position using the 2D camera (red circle). The error here is 5.2 mm.

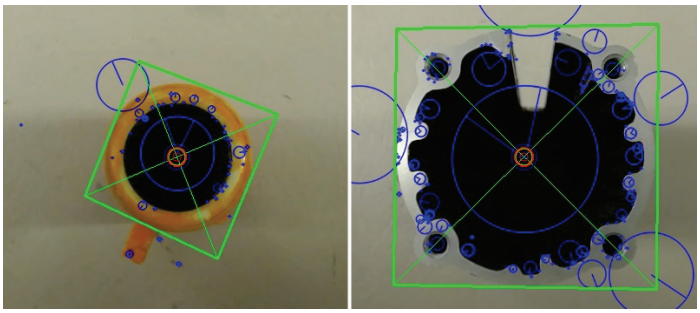
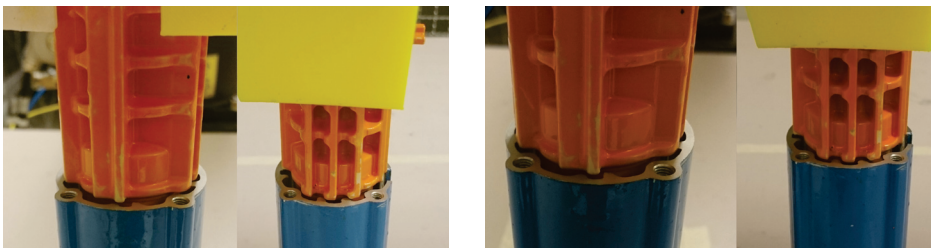


Figure 14. SIFT used on both objects to determine their position and orientation.



(**a**) (**b**)

Figure 15. Depiction of the fail and success conditions. (**a**) A slight deviation in the angle and position is considered a failure; (**b**) The position and angle are considered to be correct.

With the acquired position and orientation of both objects, a pick and place operation can be performed. In seven out of 10 assembly operations, the objects were successfully assembled, with an accuracy lower than 1 mm in position and 1° in orientation. In the remaining three attempts, the orientation of object A was the limiting factor. Typically, the failure was caused by situations as illustrated in Figure 15a, where a marginal error in orientation acquisition would prevent further execution of the assembly operation. This error was both due to the lack of accuracy in the 2D computer vision system and the gripping action, which displaced the orientation of the object.

A video of the method and conducted experiment can be downloaded at [19].

5. Discussion

Experiment 1 describes the accuracy of the 3D Detection system, while Experiment 2 describes the accuracy of the 2D Alignment system.

Experiment 1 concludes that the accuracy does not meet the requirements of the assembly task, since it shows that the maximum error was 1.96 cm, while the requirement was less than 1 mm. This, however, does meet the requirements for performing 2D alignment.

Experiment 2 concluded that the 2D alignment meets the requirements of the assembly, which is below 1°. These results are assuming that the camera is watching the object from the top, which is possible given the rough position estimate from the 3D Detection system.

Experiment 3 concludes that combining a 2D and 3D vision system, where both systems lack sufficient accuracy, can achieve said accuracy if combined.

In the experiment, it is assumed that the objects that are to be detected are not occluded and that they are standing so that the top of the object is always facing upwards. The given case assumes that the objects are properly aligned on the table, and that any irregularities, such as a fallen over or missing object, does not occur.

An extension of the system where the 2D camera can move more freely and view the object from different sides will be able to handle the special cases, where an object has an irregular alignment.

6. Conclusions

The 3D detection method resulted in an estimate with roughly 2 cm accuracy. By combining this with the 2D eye-in-hand camera to fine-tune the estimates, the accuracy was corrected to 1 mm in position and 1° in orientation. The results obtained from testing the full solution shows that such a detection system is viable in an automated assembly application.

Acknowledgments: There were no funding for the study.

Author Contributions: A.B., A.L.K. and K.L. conceived and designed the experiments; A.B. and K.L. performed the experiments; A.B. and K.L. analyzed the data; A.L.K. and O.E. contributed reagents/materials/analysis tools; A.L.K. wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dietz, T.; Schneider, U. Programming System for Efficient Use of Industrial Robots for Deburring in SME Environments. In Proceedings of the 7th German Conference on Robotics ROBOTIK 2012, Munich, Germany, 21–22 May 2012; pp. 428–433.
2. Freeman, W.T. Where computer vision needs help from computer science. In Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), San Francisco, CA, USA, 23–25 January 2011; pp. 814–819.
3. Blake, A.; Brelstaff, G. Geometry from Specularities. In Proceedings of the Second International Conference on Computer Vision, Tampa, FL, USA, 5–8 December 1988.
4. Chaumette, F.; Hutchinson, S.A. Visual Servo Control, Part II: Advanced Approaches. *IEEE Robot. Autom. Mag.* **2007**, *14*, 109–118.
5. Campbell, R.J.; Flynn, P.J. A Survey of Free-Form Object Representation and Recognition Techniques. *Comput. Vis. Image Underst.* **2001**, *210*, 166–210.
6. Khoshelham, K. Accuracy Analysis of Kinect Depth Data. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2012**, *XXXVIII-5*, 133–138.
7. Nguyen, D.D.; Ko, J.P.; Jeon, J.W. Determination of 3D object pose in point cloud with CAD model. In Proceedings of the 2015 21st Korea-Japan Joint Workshop on Frontiers of Computer Vision, Mokpo, Korea, 28–30 January 2015.
8. Luo, R.C.; Kuo, C.W.; Chung, Y.T. Model-based 3D Object Recognition and Fetching by a 7-DoF Robot with Online Obstacle Avoidance for Factory Automation. In Proceedings of the IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 26–30 May 2015; Volume 106, pp. 2647–2652.

9. Lutz, M.; Stampfer, D.; Schlegel, C. Probabilistic object recognition and pose estimation by fusing multiple algorithms. In Proceedings of the IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013; pp. 4244–4249.
10. Rennie, C.; Shome, R.; Bekris, K.E.; De Souza, A.F. A Dataset for Improved RGBD-Based Object Detection and Pose Estimation for Warehouse Pick-and-Place. *IEEE Robot. Autom. Lett.* **2016**, *1*, 1179–1185.
11. Aldoma, A.; Vincze, M.; Blodow, N.; Gossow, D.; Gedikli, S.; Rusu, R.B.; Bradski, G. CAD-model recognition and 6DOF pose estimation using 3D cues. In Proceedings of the IEEE International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 585–592.
12. Schnabel, R.; Wahl, R.; Klein, R. Efficient RANSAC for point-cloud shape detection. *Comput. Graph. Forum* **2007**, *26*, 214–226.
13. Chen, Y.; Medioni, G. Object modeling by registration of multiple range images. In Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA, USA, 9–11 April 1991; Volume 10, pp. 2724–2729.
14. Besl, P.; McKay, N. A Method for Registration of 3-D Shapes. *Int. Soc. Opt. Photonics* **1992**, 586–606, doi:10.1117/12.57955.
15. Arun, K.S.; Huang, T.S.; Blostein, S.D. Least-squares fitting of two 3-D point sets. *IEEE Trans. Pattern Anal. Mach. Intell.* **1987**, *5*, 698–700.
16. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110.
17. Rusu, R.B.; Blodow, N.; Beetz, M. Fast Point Feature Histograms (FPFH) for 3D registration. In Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 3212–3217.
18. Bay, H.; Tuytelaars, T.; Gool, L.V. Surf: Speeded up robust features. In Proceedings of the European Conference on Computer Vision, Graz, Austria, 7–13 May 2006.
19. Kleppe, A.L.; Bjørkedal, A.; Larsen, K.; Egeland, O. Website of NTNU MTP Production Group. Available online: https://github.com/ps-mtp-ntnu/Automated-Assembly-using-3D-and-2D-Cameras/ (accessed on 31 March 2017).

9.4 Paper 4: Initial Alignment of point clouds using Motors

By Adam Leon Kleppe, Lars Tingelstad and Olav Egeland

This paper presents a global initial alignment method using conformal geometric algebra. It performs similar to principal component analysis based methods, and shows that it is possible to describe point clouds using conformal geometric algebra, in ways that is more challenging with linear algebra.

The experiments show that sub-millimetre accuracy is possible, but that some considerations have to be made in order for this to be possible.

Initial Alignment of Point Clouds using Motors

Adam Leon Kleppe
Department of Mechanical and Industrial Engineering, Norwegian University of Science and Technology
N-7491
Trondheim, Norway
adam.l.kleppe@ntnu.no

Lars Tingelstad
Department of Mechanical and Industrial Engineering, Norwegian University of Science and Technology
N-7491
Trondheim, Norway
lars.tingelstad@ntnu.no

Olav Egeland
Department of Mechanical and Industrial Engineering, Norwegian University of Science and Technology
N-7491
Trondheim, Norway
olav.egeland@ntnu.no

ABSTRACT

This paper presents an approach for initial alignment or coarse registration of a partial 3D point cloud of objects. The method is based on computing the centroid of the points in the point cloud, and a line derived from the surface normals. This approach uses conformal geometric algebra and non-linear least squares optimization to achieve the results. The method performs well in experiments, and it is shown that it performs more accurately the more points are sampled.

CCS CONCEPTS

•**Mathematics of computing** → *Nonlinear equations;* •**Computing methodologies** → *Optimization algorithms;*

KEYWORDS

computer vision, conformal geometric algebra, initial alignment, screw theory

ACM Reference format:
Adam Leon Kleppe, Lars Tingelstad, and Olav Egeland. 2017. Initial Alignment of Point Clouds using Motors. In *Proceedings of CGI '17, Yokohama, Japan, June 27-30, 2017,* 5 pages.
DOI: 10.1145/3095140.3097282

1 INTRODUCTION

The 3D-3D registration problem [13] is a well-known problem in computer vision, and it is still a challenging problem. The problem is formulated as such: Assume that there are two sets of points, or point clouds

$$\boldsymbol{A} = \{\boldsymbol{a}_i\}, \quad \boldsymbol{a}_i \in \mathbb{R}^{3,1}, i = 1, \dots, m$$
$$\boldsymbol{B} = \{\boldsymbol{b}_j\}, \quad \boldsymbol{b}_j \in \mathbb{R}^{3,1}, j = 1, \dots, n$$

Find the rotation $\boldsymbol{R} \in SO(3)$ and the translation $\boldsymbol{t} \in \mathbb{R}^3$ that gives the most optimal alignment between the two sets.

$$\min \sum_i \parallel \boldsymbol{b}_{j^*} - \boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{t} \parallel^2$$

CGI '17, Yokohama, Japan

DOI: 10.1145/3095140.3097282

where $\boldsymbol{b}_{j^*}$ is the optimal corresponding point to $\boldsymbol{a}_i$, based on distance given the optimal $\boldsymbol{R}$ and $\boldsymbol{t}$.

This problem can be divided into two sub-problems. The first is to find the pose that aligns the two point clouds, and the second is to find point-wise correspondences between the sets, or in other words, which point in $\boldsymbol{A}$ corresponds to the points in $\boldsymbol{B}$, or even if there is a correspondence.

A large number of methods have been proposed to solve the registration problem [14, 19]. The most popular approach is ICP [2, 5, 15]. 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. Here the coarse registration aims to find a rough initial alignment which improves the initial conditions for the fine registration.

Most of the fine registration methods, including ICP, are so called Expectation-Maximization algorithms[13], because they alternate between solving the two sub-problems until both reaches a local minima. A known restriction with EM-algorithms is that they only find local optimal solutions. This means that in order for them to converge to the global optimum, the algorithm either has to be expanded to include global optimization techniques, such as Go-ICP [23] or Sparse ICP [3], or it has to have good initial conditions, i.e. the point clouds have to have a good initial alignment relative to each other, in order to converge to the correct solution. This is achieved with coarse registration methods, such as [7, 16–18].

The coarse registration methods usually only solves one of the sub-problems: Finding the pose that aligns the two point clouds. This means that the methods does not take the point correspondences into account. The most common approach is to create a set of features or signatures in each point cloud, and search for correspondences between the features. Examples of this are Point Signatures [6], Spin Images [12], Point Feature Histograms [16, 17], and Principal Component Analysis [7].

Both Point Signatures, Spin Images and Point Feature Histograms use techniques that originated from 2D computer vision. They use some measurement using relative distances and angles to generate features. These measurements are calculated from the points and surface normals of the point cloud. The methods uses different schemes to categorize the features, be it sets, tables or histograms to group them into cells. The main drawback with these methods is that the accuracy of the them depend on the resolution of these cells.

In this paper we propose a new method for initial alignment of two point clouds, i.e. coarse registration. The method constructs a feature using the centroid and a line computed from the surface normals of a point cloud. This feature is calculated using Conformal

Geometric Algebra. The motor which aligns the features of two point clouds is found using Non-Linear Least Square Optimization, and results in an initial alignment pose between the two point clouds. The benefit of optimization techniques to find the pose is that the accuracy does not depend on a given resolution. This method also has the benefit of requiring less computation than the methods mentioned above.

This paper is is organised as follows: Section 2 is the preliminaries, which introduces the parts of Conformal Geometric Algebra used in the paper. Section 3 describes the proposed method. Section 4 show the conducted experiment and the results, and lastly the conclusion is found in Section 5.

2 PRELIMINARIES

2.1 Conformal Geometric Algebra

The geometric algebra of the Euclidean space $\mathbb{R}^3$ is denoted $\mathbb{R}_3$, while the conformal model of geometric algebra is denoted $\mathbb{R}_{4,1}$ resulting in the null basis $\{e_0, e_1, e_2, e_3, e_\infty\}$ [9, 10]. The basis vector e_∞ represents the point at infinity, while e_0 represents an arbitrary origin. These basis vectors have the properties $e_\infty^2 = e_0^2 = 0$ and $e_\infty \cdot e_0 = -1$. The notation $\mathbb{R}_3^k$ refers to the k-grade elements of $\mathbb{R}_3$. The highest grade element of $\mathbb{R}_3$, is the Euclidean pseudoscalar, which is denoted $\boldsymbol{I}_3$. The conformal pseudoscalar is denoted $\boldsymbol{I}$. The conformal dual of a multivector $\boldsymbol{X}$ is denoted $\boldsymbol{X}^* = \boldsymbol{X}\boldsymbol{I}^{-1}$.

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

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

A line $\boldsymbol{\ell} \in \mathbb{R}^3_{4,1}$ is constructed as the outer product of two conformal points and the point at infinity:

$$\boldsymbol{\ell} = P_A \wedge P_B \wedge e_\infty.$$

This can be expressed as

$$\boldsymbol{\ell} = (\boldsymbol{p} + e_0) \wedge \hat{\boldsymbol{n}} \wedge e_\infty$$

where $\boldsymbol{p}$ is the Euclidean point and $\hat{\boldsymbol{n}}$ is the Euclidean directional vector of the line. This is called the direct representation in [9], and the OPNS representation in [10].

The dual representation of a line in conformal space is

$$\boldsymbol{\ell}^* = \boldsymbol{A} + \boldsymbol{b}e_\infty$$

where $\boldsymbol{A} = \hat{\boldsymbol{n}}^*$ is the directional bivector, and $\boldsymbol{b}$ is the momentum of the line. It is noted that $\boldsymbol{A} \wedge \boldsymbol{b} = 0$.

A screw $\boldsymbol{S}$, is a line with a pitch, meaning that

$$\boldsymbol{S}^* = \boldsymbol{A} + \boldsymbol{b}e_\infty$$

where $\boldsymbol{A} \wedge \boldsymbol{b}$ can be an arbitrary number.

A screw can be further described as

$$\boldsymbol{S}^* = \boldsymbol{A} + (\boldsymbol{b}_\parallel + \boldsymbol{b}_\perp)e_\infty$$

where $\boldsymbol{A} \wedge \boldsymbol{b}_\parallel = 0$, meaning that $\boldsymbol{A} + \boldsymbol{b}_\parallel e_\infty$ is a dual line and $\boldsymbol{b}_\perp e_\infty$ is the pitch.

A screw is generated by adding two lines together. This is shown when

$$\boldsymbol{S}^* = \boldsymbol{\ell}_1^* + \boldsymbol{\ell}_2^* = \boldsymbol{A}_1 + \boldsymbol{A}_2 + (\boldsymbol{b}_1 + \boldsymbol{b}_2)e_\infty$$

where $\boldsymbol{A}_1 \wedge \boldsymbol{b}_1 = \boldsymbol{A}_2 \wedge \boldsymbol{b}_2 = 0$, but $\boldsymbol{A}_1 \wedge \boldsymbol{b}_2$ and $\boldsymbol{A}_2 \wedge \boldsymbol{b}_1$ cannot be guaranteed zero. This also holds for the addition of multiple lines.

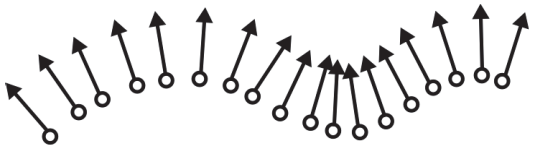


Figure 1: 2D Representation of the points in a point cloud and their respective surface normal

A flag [20] can be written as the sum of a line and a conformal point

$$\boldsymbol{f} = \boldsymbol{\ell} + \boldsymbol{P}$$

3 METHOD

This method uses two point clouds, which represents the surface of a given object, and finds the motor $\boldsymbol{M}$ which is an optimal alignment between the two point clouds. This is achieved by finding the motor which minimizes the error between the centroids of each point cloud, which results in an optimal translation, and at the same time minimizes the deviation of the average line.

Assume that two point clouds $\boldsymbol{X}$ and $\boldsymbol{Y}$ are given by a set of points and their respective surface normals.

$$\boldsymbol{X} = \{\boldsymbol{x}_i, \hat{\boldsymbol{n}}_{x_i}\}, \quad \boldsymbol{x}_i, \hat{\boldsymbol{n}}_{x_i} \in \mathbb{R}^{3,1}, ||\hat{\boldsymbol{n}}_{x_i}||_2 = 1, i = 1, \ldots, m$$
$$\boldsymbol{Y} = \{\boldsymbol{y}_j, \hat{\boldsymbol{n}}_{y_j}\}, \quad \boldsymbol{y}_j, \hat{\boldsymbol{n}}_{y_j} \in \mathbb{R}^{3,1}, ||\hat{\boldsymbol{n}}_{y_j}||_2 = 1, j = 1, \ldots, n$$

Note that the number of points in $\boldsymbol{X}$ and $\boldsymbol{Y}$ are not the same, and that $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ do not necessarily correspond if $i = j$. It is assumed that the surface normals are either calculated from the CAD model, or by estimating it using the points in the point cloud data.

3.1 Centroid

The conformal centroid of each point cloud is found by

$$\bar{\boldsymbol{P}}_X = C\left(\frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x}_i\right)$$
$$\bar{\boldsymbol{P}}_Y = C\left(\frac{1}{n}\sum_{j=1}^{n}\boldsymbol{y}_j\right)$$

where $C(\boldsymbol{p})$ is the conformal point based on the Euclidean vector $\boldsymbol{p}$.

3.2 Average of lines

The average of a set of lines is constructed as a screw which is the sum of all lines, where each line is generated from a point and its surface normal.

$\boldsymbol{L}_X$ and $\boldsymbol{L}_Y$ are the sets containing all lines generated from the point cloud.

$$\boldsymbol{L}_X = \{\boldsymbol{\ell}_{x_i} = (\boldsymbol{x}_i + n_0) \wedge \hat{\boldsymbol{n}}_{x_i} \wedge n_\infty\}$$
$$\boldsymbol{L}_Y = \{\boldsymbol{\ell}_{y_j} = (\boldsymbol{y}_j + n_0) \wedge \hat{\boldsymbol{n}}_{y_i} \wedge n_\infty\}$$

The sum of these lines become a screw, where the screw axis is computed as the average of the lines. The point cloud, generated lines, average line and centroid can be viewed in Figure 2. It can be

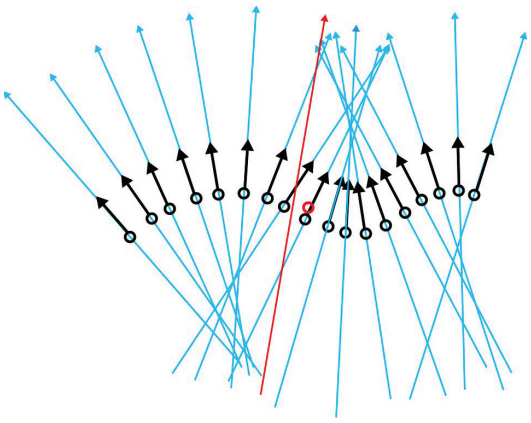


Figure 2: Lines generated from each point in the point cloud. The red line represents the average line, while the red point is the centroid

seen here that the screw axis does not necessarily pass through the centroid.

The screws of the two point clouds, X and Y, are defined as

$$S_X = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\ell}^*_{x_i}$$

$$S_Y = \frac{1}{m}\sum_{j=1}^{m} \boldsymbol{\ell}^*_{y_j}$$

It can be seen that these screws are the average of all lines since

$$\begin{aligned} S &= \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{\ell}^*_i \\ &= \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{A_i} + \boldsymbol{b}_i e_\infty \\ &= \frac{1}{n}\left(\sum_{i=1}^{n} \boldsymbol{A_i} + \sum_{i=1}^{n} \boldsymbol{b}_i e_\infty\right) \\ &= \bar{\boldsymbol{A}} + \bar{\boldsymbol{b}} e_\infty \end{aligned}$$

where $\bar{\boldsymbol{A}}$ is the average directional bivector, and $\bar{\boldsymbol{b}}$ is the average momentum. This can be written as

$$S = \bar{\boldsymbol{A}} + (\bar{\boldsymbol{b}}_{\parallel} + \bar{\boldsymbol{b}}_{\perp}) e_\infty$$

where $\bar{\boldsymbol{A}} \wedge \bar{\boldsymbol{b}}_{\parallel} = 0$.

This can be rewritten using the average line $\bar{\boldsymbol{\ell}}$, which is the average of the combined lines.

$$\begin{aligned} S &= \bar{\boldsymbol{A}} + (\bar{\boldsymbol{b}}_{\parallel} + \bar{\boldsymbol{b}}_{\perp}) e_\infty \\ &= \bar{\boldsymbol{A}} + \bar{\boldsymbol{b}}_{\parallel} e_\infty + \bar{\boldsymbol{b}}_{\perp} e_\infty \\ &= \bar{\boldsymbol{\ell}}^* + \bar{\boldsymbol{b}}_{\perp} e_\infty \end{aligned}$$

3.3 Restrictions

There are in total 7 degrees of freedom in a motion, 3 to translation, 3 to rotation and 1 to scale [11]. This method uses only one line and one point is known in both point clouds and therefore in both coordinate systems. This forces seven constraints upon the system, 4 from the line and 3 from the point. This means the method is able to perform any rigid-body motion.

Since each line is generated by a point $\boldsymbol{p}$ and its corresponding surface normal $\hat{\boldsymbol{n}}$, these lines can be described as a force $\boldsymbol{F} = \hat{\boldsymbol{n}}$ at $\boldsymbol{p}$, where $|F| = 1$. To sum all these forces is the same as evaluating the force applied over the whole surface area, which by definition is the same as pressure.

This forces a restriction upon the method: This method cannot be used on the whole surface of an object. This is because the pressure over an enclosed surface area is zero. This means that if the whole surface is sampled by points and these points generate lines, then the sum of these lines will be zero.

As mentioned, this method is used on point clouds generated from the viewable surfaces of an object. In practice, this means that this restriction will never occur, since the whole surface of an object cannot be viewed at the same time.

3.4 Motor Estimation

Each point cloud, X and Y, have one centroid, $\bar{P}_X$ and $\bar{P}_Y$, and one screw axis, $\bar{\boldsymbol{\ell}}_X$ and $\bar{\boldsymbol{\ell}}_Y$.

From these, their respective flags are defined as

$$f_X = \bar{\boldsymbol{\ell}}_X + \bar{P}_X$$
$$f_Y = \bar{\boldsymbol{\ell}}_Y + \bar{P}_Y$$

In order to minimize the error between these two flags, we can find the optimal motor between them.

$$\begin{aligned} & f_X - \boldsymbol{M}^\dagger f_Y \tilde{\boldsymbol{M}}^\dagger \\ & \bar{\boldsymbol{\ell}}_X - \boldsymbol{M}^\dagger \bar{\boldsymbol{\ell}}_Y \tilde{\boldsymbol{M}}^\dagger + \bar{P}_X - \boldsymbol{M}^\dagger \bar{P}_Y \tilde{\boldsymbol{M}}^\dagger = 0 \end{aligned}$$

where $\boldsymbol{M}^\dagger$ is the optimal motor between the two point clouds. The result of zero is only possible if both point clouds are identical.

When comparing two point clouds which are not identical, either because of added noise or different points are used, an optimization scheme could be used. The motor which is the optimal transform between the two flags, will have one unique solution, since two lines and two points are used in the minimization function [4, 8].

3.5 Error Functions

The optimization of the motor between the two centroids in the flag only require an error measurement in the form of distance. The distance measure between the two centroids is easily found with

$$\epsilon_P = d^2 = \bar{P}_X \cdot \boldsymbol{M} \bar{P}_Y \tilde{\boldsymbol{M}}$$

where ϵ_P is the error function for the two centroids.

For the average lines in the flag, both the distance and the angle has to be optimized [1]. These parameters can be extracted from the motor $\boldsymbol{M}$which transforms one to the other.

$$\boldsymbol{M} = \frac{\bar{\boldsymbol{\ell}}_X}{\bar{\boldsymbol{\ell}}_Y}$$

where

$$\boldsymbol{M} = \cos\theta - \sin\theta \boldsymbol{A}_n - \sin\theta \boldsymbol{b}_n e_\infty - d\cos\theta \boldsymbol{a}_n e_\infty + d\sin\theta e_1 e_2 e_3 e_\infty$$

where $\boldsymbol{A}_n = \boldsymbol{a}_n^*$ and $\boldsymbol{b}_n$ are components of the common normal line $\boldsymbol{\ell}_n$.

According to [22], a good error function for angles is $\sin\frac{\theta}{2}$. This is because the error function is at most $\sin\frac{\pi}{2} = 1$, which means that outliers do not have a large error, but are bound by 1. This can be calculated using

$$\frac{1}{2}(1-\cos\theta) = \sin^2\frac{\theta}{2}$$

The distance between the two lines can be calculated by decomposing the motor in a different manner:

$$\boldsymbol{M} = \boldsymbol{T}\boldsymbol{R}$$

where

$$\boldsymbol{T} = 1 - \frac{1}{2}\boldsymbol{t}e_\infty$$
$$\boldsymbol{R} = -e_0 \cdot (\boldsymbol{M}e_\infty)$$

where $\boldsymbol{T}$ is the translation and $\boldsymbol{R}$ is the rotation.

The distance between the lines can then be calculated using

$$\delta = \| \boldsymbol{\delta} \|$$
$$\boldsymbol{\delta} = \frac{\boldsymbol{t} \wedge \boldsymbol{B}}{\boldsymbol{B}}$$

where

$$\boldsymbol{t} = -2\frac{e_0 \cdot \boldsymbol{M}}{\boldsymbol{R}}$$
$$\boldsymbol{B} = \frac{\langle \boldsymbol{R} \rangle_2}{\| \langle \boldsymbol{R} \rangle_2 \|}$$

where $\langle \boldsymbol{R} \rangle_2$ is the 2-blade component of $\boldsymbol{R}$.

With these parameters, the error function $\epsilon_{\bar{\boldsymbol{\ell}}}$ can be formulated as

$$\epsilon_{\bar{\boldsymbol{\ell}}} = \delta^2 + \sin^2\frac{\theta}{2} = \delta^2 + \frac{1}{2}(1-\cos\theta)$$

3.6 Non-linear Least Square Optimization

The non-linear least square optimization solver was used to estimate the motor. By using the given error functions, the equation to be minimized is given by

$$\min \frac{1}{2}\left(\epsilon_{\bar{\boldsymbol{\ell}}} + \epsilon_{\boldsymbol{p}}\right)^2 = \min \frac{1}{2}\left(\delta^2 + \sin^2\frac{\theta}{2} + d^2\right)^2$$

such that

$$\boldsymbol{M}\tilde{\boldsymbol{M}} = 1$$

which is solved using Levenberg-Marquardt algorithm which is has been developed to work for conformal points and lines. The algorithm is presented in [21].

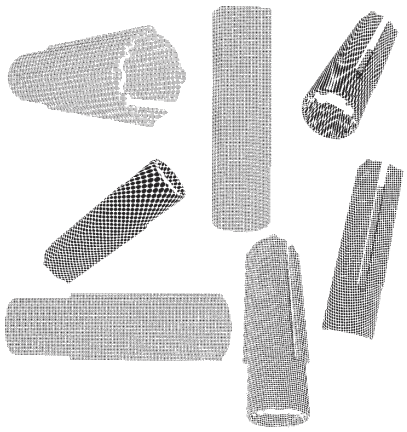


Figure 3: Seven viewpoint samples generated from one of the objects.

4 EXPERIMENTS

Several point clouds were generated from a set of CAD models. Since the method only works when having partial surfaces, the point clouds were generated using viewpoint sampling.

Viewpoint sampling was done by placing a virtual camera facing the 3D model. The points are generated based on the surfaces that are visible by the virtual camera. In order to get the whole view of the CAD model, several samples were generated from different viewpoints. The position of the viewpoints were calculated using a tessellated sphere which surrounded the 3D model. Each vertex of the tessellated sphere was set as a viewpoint.

Two models each generated 3 point clouds per viewpoint, one with 100 points, one with 1000 points and one with 10000 points. There were a total of 42 viewpoints per object, resulting in a total of 252 point clouds. A sample of these point clouds is shown in Figure 3. Each point cloud was given an arbitrary transformation and noise was applied to the point cloud. The initial alignment method together with the GAME framework was used to calculate the transformation between the two point clouds. The error was calculated by applying the Root Mean Square between the points in each point cloud, and is measured in meters.

$$\epsilon = \sqrt{\frac{1}{n} P_{X_1} \cdot \boldsymbol{M} P_{Y_2} \tilde{\boldsymbol{M}} + P_{X_2} \cdot \boldsymbol{M} P_{Y_2} \tilde{\boldsymbol{M}} \ldots P_{X_n} \cdot \boldsymbol{M} P_{Y_n} \tilde{\boldsymbol{M}}}$$

The result of these tests are shown in Table 1. It can be seen from the table that the error decreased as the number of samples increased. This is expected since the more samples are used, the more the average will cancel out the added noise. The average error was 6.8325×10^{-4} m for a 100 samples, 3.323×10^{-4} m for a 1000 samples and 3.7214×10^{-5} m for a 10000 samples.

5 CONCLUSION

This paper shows a method for initial alignment for point clouds. The method finds the optimal motor between the centroid and

	100 Samples		1000 Samples		10000 Samples	
Point Cloud	# Iteration	Error [m]	# Iteration	Error [m]	# Iteration	Error [m]
1	6	1.840×10^{-3}	6	9.4128×10^{-3}	17	6.6647×10^{-7}
6	7	4.8510×10^{-4}	16	1.3453×10^{-6}	12	3.5899×10^{-7}
7	7	3.1998×10^{-4}	14	2.5855×10^{-6}	12	2.5160×10^{-6}
14	7	1.8475×10^{-4}	7	5.7251×10^{-3}	12	1.0650×10^{-8}
22	7	1.1063×10^{-3}	6	1.2058×10^{-2}	11	3.9998×10^{-8}
31	6	1.7973×10^{-3}	7	5.5537×10^{-3}	7	1.0105×10^{-2}
45	7	1.0676×10^{-3}	13	4.3508×10^{-9}	11	8.0189×10^{-8}
65	7	1.1544×10^{-4}	13	2.5538×10^{-9}	13	8.5678×10^{-9}
73	12	8.7525×10^{-9}	11	7.6853×10^{-9}	10	4.7666×10^{-9}
78	11	1.0848×10^{-8}	11	3.4762×10^{-8}	12	6.9853×10^{-9}
Average		6.8325×10^{-4}		3.323×10^{-4}		3.7214×10^{-5}

Table 1: Sample of the result of the initial alignment method. The number of iterations before the method terminated and the resulting error from the true transform is shown. The average of all results is also shown.

average of lines of two point clouds, which is the initial alignment. The average error was 6.8325×10^{-4} m for a 100 samples, 3.323×10^{-4} m for a 1000 samples and 3.7214×10^{-5} m for a 10000 samples.

REFERENCES

[1] Eduardo Bayro-Corrochano, Kostas Daniilidis, and Gerald Sommer. 2000. Motor algebra for 3D kinematics: the case of the hand-eye calibration. *Journal of Mathematical Imaging and Vision* 13, 2 (2000), 79–100. DOI: https://doi.org/10.1023/A:1026567812984

[2] Paul Besl and Neil McKay. 1992. A Method for Registration of 3-D Shapes. (1992), 239–256 pages. DOI: https://doi.org/10.1109/34.121791

[3] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. 2013. Sparse iterative closest point. *Computer Graphics Forum* 32, 5 (2013), 113–123. DOI: https://doi.org/10.1111/cgf.12178

[4] H.H. Chen. 1991. A screw motion approach to uniqueness analysis of head-eye geometry. *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (1991), 145–151. DOI: https://doi.org/10.1109/CVPR.1991.139677 arXiv:cs/9605103

[5] Y. Chen and G. Medioni. 1991. Object modeling by registration of multiple range images. *Proceedings. 1991 IEEE International Conference on Robotics and Automation* April (1991), 2724–2729. DOI: https://doi.org/10.1109/ROBOT.1991.132043

[6] Chin Seng Chua and Ray Jarvis. 1997. Point Signatures: A New Representation for 3D Object Recognition. *International Journal of Computer Vision* 25, 1 (1997), 63–85. DOI: https://doi.org/10.1023/A:1007981719186

[7] Do Hyun Chung, Il Dong Yun, and Sang Uk Lee. 1997. Registration of Multiple Range Views using the Reverse Calibration Technique. (1997), 0–19.

[8] Kostas Daniilidis. 1999. Hand-Eye Calibration Using Dual Quaternions. *The International Journal of Robotics Research* 18, 3 (1999), 286–298. DOI: https://doi.org/10.1177/02783649922066213

[9] Leo Dorst, Daniel Fontijne, and Stephen Mann. 2009. *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA. 664 pages. http://www.geometricalgebra.net/

[10] Dietmar Hildenbrand. 2013. *Foundations of Geometric Algebra Computing*. Geometry and Computing, Vol. 8. Springer Berlin Heidelberg, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-642-31794-1

[11] Berthold K P Horn. 1987. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A* 4, 4 (1987), 629. DOI: https://doi.org/10.1364/JOSAA.4.000629

[12] Andrew E. Johnson and Martial Hebert. 1999. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21, 5 (1999), 433–449. DOI: https://doi.org/10.1109/34.765655

[13] Hongdong Li and Richard Hartley. 2007. The 3D-3D registration problem revisited. *Proceedings of the IEEE International Conference on Computer Vision* (2007). DOI: https://doi.org/10.1109/ICCV.2007.4409077

[14] Helmut Pottmann, Stefan Leopoldseder, and Michael Hofer. 2004. Registration without ICP. *Computer Vision and Image Understanding* 95, 1 (2004), 54–71. DOI: https://doi.org/10.1016/j.cviu.2004.04.002

[15] S. Rusinkiewicz and M. Levoy. 2001. Efficient variants of the ICP algorithm. *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM* 2001-Janua (2001), 145–152. DOI: https://doi.org/10.1109/IM.2001.924423

[16] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. 2009. Fast Point Feature Histograms (FPFH) for 3D registration. *IEEE International Conference on Robotics and Automation* (2009), 3212–3217. DOI: https://doi.org/10.1109/ROBOT.2009.5152473

[17] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. 2010. Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram. (2010), 2155–2162.

[18] B Sabata and J. K. Aggarwal. 1996. Surface Correspondence and Motion Computation from a Pair of Range Images. *Computer Vision and Image Understanding* 63, 2 (1996), 232–250. DOI: https://doi.org/10.1006/cviu.1996.0017

[19] Joaquim Salvi, Carles Matabosch, David Fofi, and Josep Forest. 2007. A review of recent range image registration methods with accuracy evaluation. *Image and Vision Computing* 25, 5 (2007), 578–596. DOI: https://doi.org/10.1016/j.imavis.2006.05.012

[20] J. M. Selig. 2005. *Geometric Fundamentals of Robotics*. 398 pages. DOI: https://doi.org/10.1007/b138859 arXiv:arXiv:1011.1669v3

[21] Lars Tingelstad and Olav Egeland. 2016. Motor Estimation using Heterogeneous Sets of Objects in Conformal Geometric Algebra. *Advances in Applied Clifford Algebras* (2016), 1–15. DOI: https://doi.org/10.1007/s00006-016-0692-8

[22] Robert Valkenburg and Leo Dorst. 2011. Estimating Motors from a Variety of Geometric Data in 3D Conformal Geometric Algebra. *Guide to Geometric Algebra in Practice* XVII, December (2011), 25–45. DOI: https://doi.org/10.1007/978-0-85729-811-9 arXiv:arXiv:1011.1669v3

[23] Jiaolong Yang, Hongdong Li, and Yunde Jia. 2013. Go-ICP: Solving 3D registration efficiently and globally optimally. *Proceedings of the IEEE International Conference on Computer Vision* (2013), 1457–1464. DOI: https://doi.org/10.1109/ICCV.2013.184

9.5 Paper 5: Coarse Alignment for Model Fitting of Point Clouds using a Curvature-Based Descriptor

By Adam Leon Kleppe, Lars Tingelstad and Olav Egeland

This paper presents the curvature-based descriptor, which uses conformal geometric algebra. Through the experiments it shows that it can achieve a sub-millimeter accuracy, which is not achieved with the other descriptors in the experiments.

Note that in the paper, the conformal geometric algebra does not mention the dual terms, from Section 4.1.6. This was intentional, since adding these terms and calculations would make the paper unnecessary complex. This means that the planes and spheres presented in the paper are actually the duals of those represented in the thesis.

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 [20] 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 [30], [5], where Iterative Closest Point (ICP) [3], [6], [25] 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 [20], 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 [37] 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], [26], [27], [29].

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 [18] and Point Feature Histograms [26], [27].

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], [18], [27], [29] 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 [22], [17], the curvature of an object can be used to calculate different properties of the object. In [22], it is used for 2D alignment, while in [17] 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 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 [26], 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 $\mathbb{R}^{4,1} = \operatorname{span}\{e_1, e_2, e_3, e_0, e_\infty\}$ 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 $[\boldsymbol{p}] = [p_1, p_2, p_3]^\mathsf{T}$. 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

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

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

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

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], [24]

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

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

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

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

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

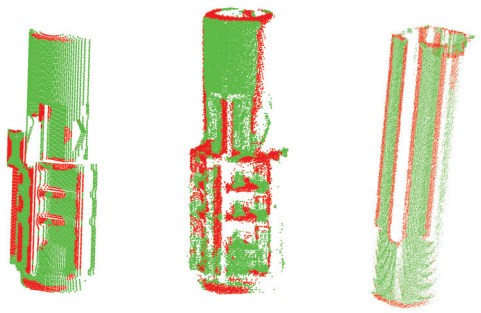
where $n_{\min}$, δ_l, δ_p and δ_s are user-specified keypoint parameters. The parameter n_{i} defines the minimum of neighbouring points that is required. This parameter is used to ensure that the calculated shape factors for $\boldsymbol{p}_i$ are reliable. With too few neighbouring points, which happens 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.

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

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



(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.

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 [23] 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

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

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

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

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

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

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 3 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.

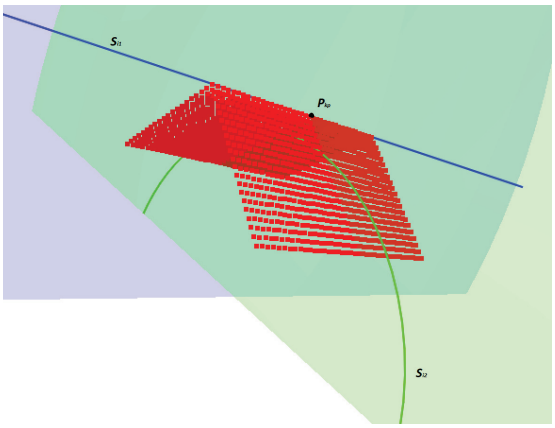


Fig. 2. An example of a descriptor. The spheres are shown as circles in the figure to make it easier to view. The blue circle of $\boldsymbol{S}_1$ lies on the blue $\mathbf{v}_1$-$\mathbf{v}_3$ plane, while the green circle of $\boldsymbol{S}_2$ lies on the green $\mathbf{v}_2$-$\mathbf{v}_3$ plane.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

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

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

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 3.

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], [35].

In this work, the pose estimation was done in terms of Conformal Geometric Algebra using the method of [36]. 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 [36] and the SVD method of [2], [35], 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 [33]

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

where $A = a_1 e_{23} + a_2 e_{31} + a_3 e_{12}$, $B = b_1 e_{23} + b_2 e_{31} + b_3 e_{12}$ 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

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

Let c_i, $i = 1, \ldots, 8$ denote the basis elements of $\mathbb{M}$, that is, $c_1 = 1, c_2 = e_{23}, c_3 = e_{31}, \ldots$. 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 $\boldsymbol{m} = [m_1, \ldots, m_8]^{\mathrm{T}}$, 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 $\mathbf{y}_k \in \boldsymbol{Y}_{\text{keypoints}}$, 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 [36]. 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

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

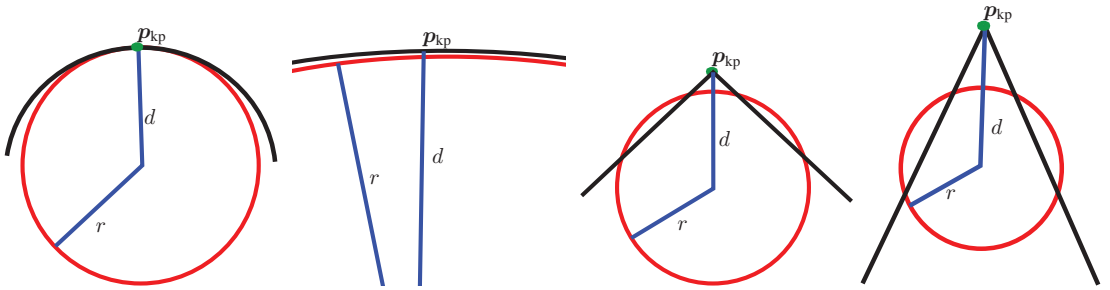
Let the basis elements of $\overline{\mathbb{M}}$ be denoted c^i, $i = 1, \ldots, 8$. Then the basis elements c^i of $\overline{\mathbb{M}}$ will be reciprocal to the basis elements c_i of $\mathbb{M}$, which means that $c^j \cdot c_i = \langle c^j c_i \rangle = \delta_{ij}$ and δ_{ij} is the Kronecker 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

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



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

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) [26], Point-Pair Features (PPF) [12], Signature of Histogram of OrienTation (SHOT) [34], 3D Shape Context (3DSC) [13] and Globally Aligned Spatial Distribution (GADS) [21].

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

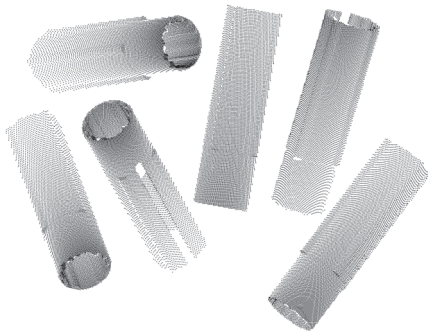


Fig. 4. Multiple point clouds of the same 3D CAD model, from multiple views

A. Setup

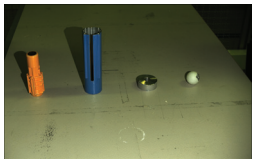
1) Hardware: The hardware that was used for the experiments, was the same for all three. The computer that was used was a desktop computer with an Intel Core i7 7700k Sky Lake at $4.2\,\mathrm{GHz}$ with 32GB $2666\,\mathrm{MHz}$ DDR4 and a EVGA GeForce GTX 1080 Founders Edition graphics card. The computer 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 [31]. The Zivid 3D camera outputs 2.3 Mpixel RGBD image, with a field of view of $425\times267\,\mathrm{mm}$ at a distance of $0.6\,\mathrm{m}$ with a depth resolution of $0.1\,\mathrm{mm}$ at the same distance.

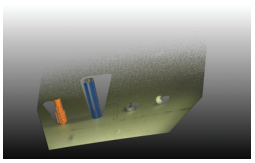
2) Point Cloud Data: There were a total of 129 point clouds used in the experiments.

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 4.

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 [32], where RANSAC

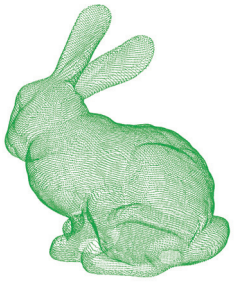


(a) 2D image taken with the Zivid Camera



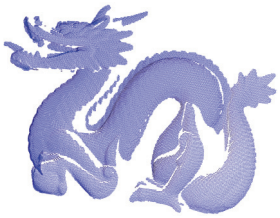
(b) Point cloud acquired by the Zivid Camera

Fig. 5. 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. 6. A sample of the database point clouds used.

was used to find the table, and the region growing algorithm was used to separate the objects. Each object was saved as an individual point cloud, resulting in 10 point clouds. A sample is shown in Fig 5.

35 point clouds were from the Stanford database [19], where 5 of them were of the Stanford Bunny, 15 of the Stanford Happy Buddha and 15 of the Stanford Dragon. Fig 6 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 [28], 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 [21] 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 7, Fig 8 and Fig 9. It is seen in the

figures that there is a correlation between the accuracy of the method and the r and w parameters.

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 pose estimation, but with the point correspondence the optimal solution ranges between 0.05 and 0.2 depending on the point cloud. 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 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 radiance.

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.

The RMS shown in Table III is calculated between the orange object shown in Fig 5b 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 $10\,\mathrm{cm}$ on average, and only by using the ICP method, did it achieve a more accurate pose. The resulting pose estimation can be seen in Fig 10

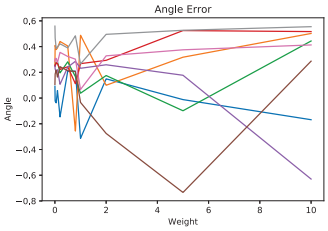
V. Discussion

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.

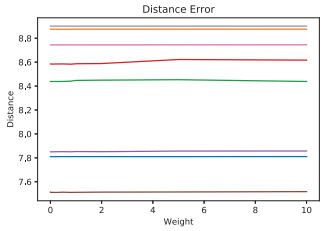
For instance, in the preprocessing step Section III-B1, each point in the point cloud checks which points are within a given radius. In the code, the method goes through every point for each point, making it an algorithm with a complexity of $O(n^2)$. This could be greatly improved by using smarter methods 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], [35]

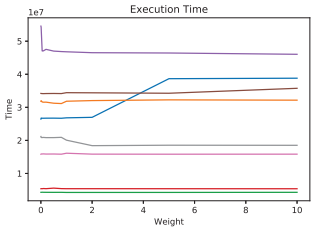
It is also worth noting that the results from the GADS method, could probably be improved, if the point clouds would have color information. This was not possible in the



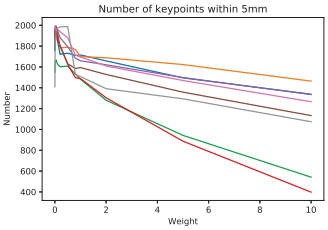
(a) Angle error [rad], measured using an angle-axis calculation of the error transformation



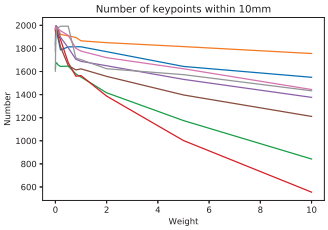
(b) Distance error [mm], measured using the distance of the error transformation



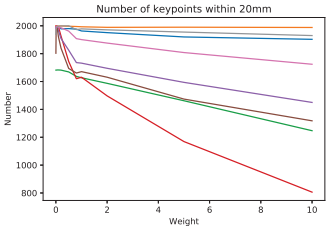
(c) Execution time [µs]



(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

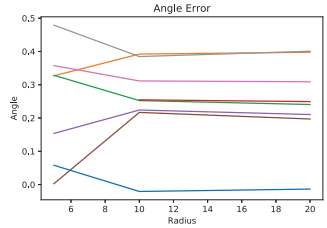
Fig. 7. Results when adjusting the weight parameter. Each coloured graph represents the results from a sample point cloud, where 8 are selected from the total of 129 point clouds. The red, green and orange graphs are of different viewpoint point clouds of part A, the blue, purple and pink graphs are 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.

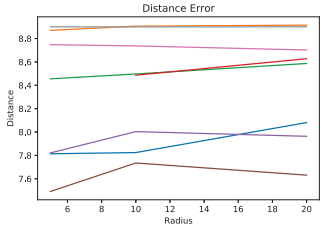
	Noise $\sigma = 0.1$		Noise $\sigma = 0.3$		Noise $\sigma = 0.5$		Noise $\sigma = 1.0$		Noise $\sigma = 2.0$	
	Angle [rad]	Distance [mm]	Angle [rad]	Distance [mm]	Angle [rad]	Distance [mm]	Angle [rad]	Distance [mm]	Angle [rad]	Distance [mm]
Proposed	0.122	0.645	0.132	0.833	0.151	0.944	0.262	1.267	0.643	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.

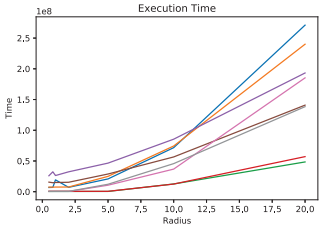
	Noise $\sigma = 0.1$	Noise $\sigma = 0.3$	Noise $\sigma = 0.5$	Noise $\sigma = 1.0$	Noise $\sigma = 2.0$
	Execution Time [ms]	Execution Time [ms]	Execution Time [ms]	Execution Time [ms]	Execution Time [ms]
Proposed	2329.517	2328.523	2330.665	2430.101	2354.512
FPFH	829.594	843.941	901.421	830.512	854.442
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



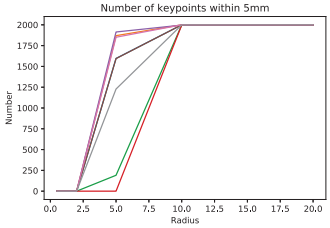
(a) Angle error [rad], measured using an angle-axis calculation of the error transformation



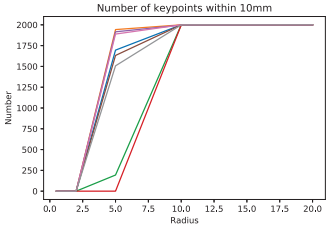
(b) Distance error [mm], measured using the distance of the error transformation



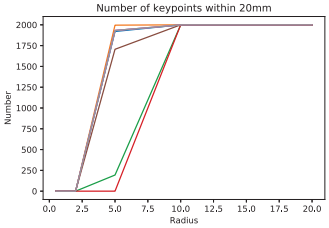
(c) Execution time [µs]



(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. 8. 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.

TABLE III. RESULTS FROM EXPERIMENT 3. THE RMS SHOWS THE RMS BETWEEN THE ICP CORRESPONDING POINTS AFTER 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	2.56153	21487.476	
FPFH	2.56155	2390.431	Point cloud was flipped upside down
PPF	2.56153	56926.563	Estimate before ICP was far away from the actual point cloud
SHOT	2.74457	30411.992	Got a better RMS with a different point cloud
3DSC	2.56153	163224.032	
GADS	3.11244	1224.032	The orientation was not correct

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 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.

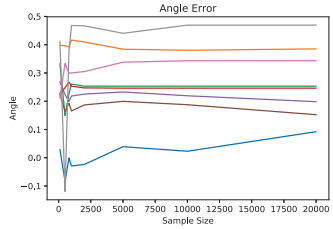
[3] P. Besl and N. McKay. A Method for Registration of 3-D Shapes, 1992.

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

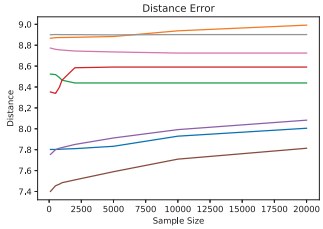
[5] U. Castellani and A. Bartoli. 3D shape registration. *3D Imaging, Analysis and Applications*, 9781447140:221–264, 2014.

[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.

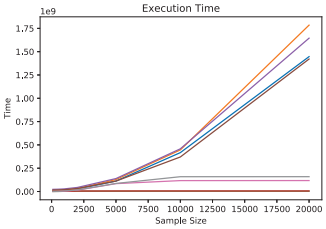
[7] C. S. Chua and R. Jarvis. Point Signatures: A New Representation for 3D Object Recognition. *International Journal of Computer Vision*, 25(1):63–85, 1997.



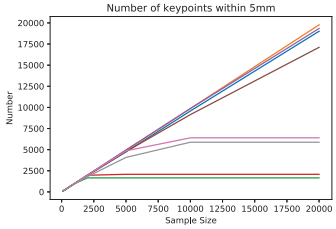
(a) Angle error [rad], measured using an angle-axis calculation of the error transformation



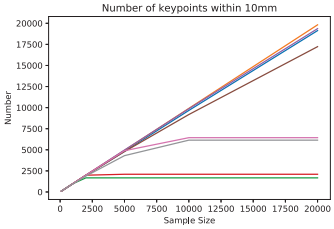
(b) Distance error [mm], measured using the distance of the error transformation



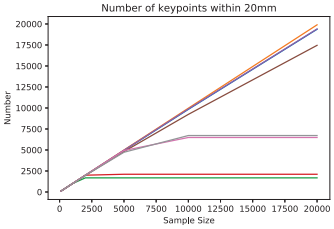
(c) Execution time [µs]



(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 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.

[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] 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.

[18] 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.

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

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

[21] 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.

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

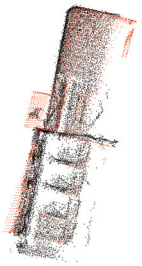
[23] V. Pratt. Direct least-squares fitting of algebraic surfaces. *ACM SIGGRAPH Computer Graphics*, 21(4):145–152, 1987.

[24] 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.

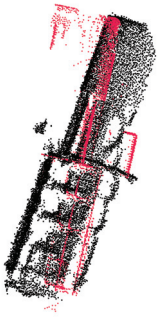
[25] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM*, 2001-Janua:145–152, 2001.

[26] 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.

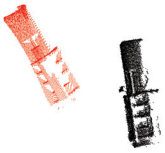
[27] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3D Recognition



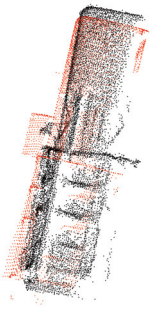
(a) Results from the curvature-based descriptor.



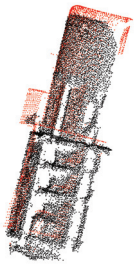
(b) Results from FPFH.



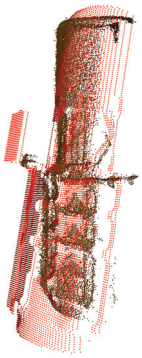
(c) Results from PPF.



(d) Results from SHOT.



(e) Results from 3DSC.



(f) Results from GASD.

Fig. 10. 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.

and Pose Using the Viewpoint Feature Histogram. In *Intelligent Robots and Systems (IROS),*, pages 2155–2162, 2010.

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

[29] 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.

[30] 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.

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

[32] 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.

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

[34] 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.

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

[36] 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.

[37] 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.

9.6 Paper 6: A Curvature-Based Descriptor for Point Cloud Alignment using Conformal Geometric Algebra

By Adam Leon Kleppe, Lars Tingelstad and Olav Egeland

This paper presents an improvement on the descriptor presented in Paper 5. This takes a deeper look into the point correspondence method, and improves it by using a geometric interpretation of the descriptor.

The experiments show that the descriptor performs better than many of the other descriptors presented in this thesis.

Is not included due to copyright

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. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1983.

[3] D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[5] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[6] P. Besl and N. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[7] P. J. Besl and R. C. Jain. Segmentation Through Variable-Order Surface Fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2):167–192, 1988.

[8] T. Birdal and S. Ilic. A Point Sampling Algorithm for 3D Matching of Irregular Geometries. *Iros*, pages 6871–6878, 2017.

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

[10] R. Bro, E. Acar, and T. Kolda. Singular Value Decomposition (lit) Sandia National Laboratories. *Sandia report*, 1(October):135–140, 2007.

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

[12] M.-c. Chang, F. F. Leymarie, and B. B. Kimia. 3D shape registration using regularized medial scaffolds. In *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pages 987–994, 2004.

[13] 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.

[14] N. Chernov and C. Lesort. Least squares fitting of circles. *Journal of Mathematical Imaging and Vision*, 23:1–239, 2005.

[15] 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.

[16] M. Corsini, P. Cignoni, and R. Scopigno. Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):914–924, 6 2012.

[17] P. Cunningham and S. J. Delany. k-Nearest Neighbour Classifiers. *Multiple Classifier Systems*, 34(APRIL 2007):1–17, 2007.

[18] 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.

[19] 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.

[20] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3D object recognition. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, July:998–1005, 6 2010.

[21] M. a. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analsis and Automated Carography. *Communications of the ACM*, 24(6):381–395, 1981.

[22] 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.

[23] J. Garstka and G. Peters. Fast and Robust Keypoint Detection in Unstructured 3-D Point Clouds. *12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, pages 131–140, 2015.

[24] E. Grilli, F. Menna, and F. Remondino. A review of point clouds segmentation and classification algorithms. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 42(2W3):339–344, 2017.

[25] C. Harris and M. Stephens. A Combined Corner and Edge Detector. *Proceedings of the Alvey Vision Conference 1988*, pages 1–23, 1988.

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

[27] D. Hildenbrand, E. Bayro-Corrochano, and J. Zamora. Advanced geometric approach for graphics and visual guided robot object manipulation. *Proceedings - IEEE International Conference on Robotics and Automation*, 2005:4727–4732, 2005.

[28] 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.

[29] A. Kleppe, L. Tingelstad, and O. Egeland. Initial alignment of point clouds using motors. *ACM International Conference Proceeding Series*, Part F1286, 2017.

[30] A. L. Kleppe, A. Bjørkedal, K. Larsen, and O. Egeland. Automated assembly using 3D and 2D cameras. *Robotics*, 6(3):14, 2017.

[31] A. L. Kleppe and O. Egeland. Inverse Kinematics for Industrial Robots using Conformal Geometric Algebra. *Modeling, Identification and Control: A Norwegian Research Bulletin*, 37(1):63–75, 2016.

[32] 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.

[33] A. L. Kleppe, L. Tingelstad, and O. Egeland. Coarse Alignment for Model Fitting of Point Clouds using a Curvature-Based Descriptor. *IEEE Transaction on Automation Science and Engineering*, 2018.

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

[35] Z. C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz. General 3D modelling of novel objects from a single view. *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pages 3700–3705, 2010.

[36] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[37] N. J. Mitra and A. Nguyen. Estimating surface normals in noisy point cloud data. *Proceedings of the nineteenth conference on Computational geometry - SCG '03*, page 322, 2003.

[38] V. Murino, L. Ronchetti, U. Castellani, and A. Fusiello. Reconstruction of complex environments by robust pre-aligned ICP. In *Proceedings of International Conference on 3-D Digital Imaging and Modeling, 3DIM*, volume 2001-Janua, pages 187–194, 2001.

[39] D. Nehab and P. Shilane. Stratified Point Sampling of 3D Models. *Eurographics Symposium on Point-Based Graphics*, pages 49–56, 2004.

[40] C. Perwass. *Geometric Algebra with Applications in Engineering*, volume 4 of *Geometry and Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

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

[42] T. Rabbani, F. a. van den Heuvel, and G. Vosselman. Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences - Commission V Symposium 'Image Engineering and Vision Metrology'*, 36(5):248–253, 2006.

[43] G. Rauscher, D. Dube, and A. Zell. A Comparison of 3D Sensors for Wheeled Mobile Robots. *Intelligent Autonomous Systems*, 392(13):29–41, 2016.

[44] M. Ritter, H. Moritsch, and W. Leimer. Visual Data Mining Using the Point Distribution Tensor. In *ICONS*, pages 10–13, 2012.

[45] 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.

[46] R. B. Rusu. Semantic 3D Object Maps for Everyday Manipulation in Human. *Kunstliche Intelligenz*, 24(4):345–348, 2010.

[47] 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.

[48] 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.

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

[50] S. Salti, F. Tombari, R. Spezialetti, and L. D. Stefano. Learning a descriptor-specific 3D keypoint detector. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:2318–2326, 2015.

[51] 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.

[52] P. D. Sampson. Fitting conic sections to "very scattered" data: An iterative refinement of the bookstein algorithm. *Computer Graphics and Image Processing*, 18(1):97–108, 1982.

[53] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: modelling, planning and control*. Springer, 2009.

[54] M. W. Spong, S. A. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 2005.

[55] 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):1961–1976, 2017.

[56] 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.

[57] S. Umeyama. Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, 1991.

[58] 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.

[59] 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.

[60] 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.

[61] J. Zamora and E. Bayro-Corrochano. Inverse kinematics, fixation and grasping using conformal geometric algebra. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 4(1 1):3841–3846, 2004.