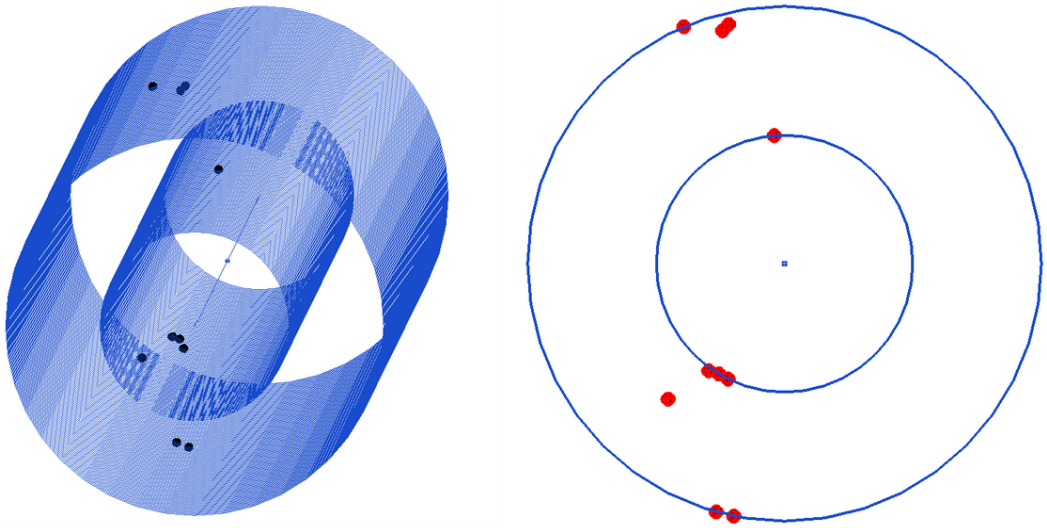
DETERMINING CYLINDER PAIRS FROM POINT CLOUDS

RUURD MOELKER



Master Thesis
May 2014

Faculty of Mathematics and Natural Sciences
University of Groningen

Supervisors:
Dr. Henk Bekker
Prof. Dr. Wim H. Hesselink

Ruurd Moelker, *Determining cylinder pairs from point clouds* , May 2014

ABSTRACT

We consider the following problem. Given a set P of N points in 3D. We aim to find the pair of coaxial cylinders, with the property that one cylinder encloses P and the other is inscribed in P, and with the property that the difference between the radii of these cylinders, called cylindricity, is minimal.

This research is done in cooperation with the company Schut Geometrical Metrology, manufacturing coordinate measuring machines primarily for use in metrology. A point set P results from measuring a part. Based on the cylindricity of P the part is accepted or rejected.

This research has two distinct goals. First we consider the problem of determining the cylinder pair from the minimal number of points. Using this result, in the second part the cylindricity of P is determined, where P consists of more than the minimal number of points. Solving sets of polynomial equations is an essential part of the first part. An important condition for the software resulting from this project is that it should not depend on any external commercial libraries or software.

ACKNOWLEDGMENTS

My graduation research has provided me with a fair share of geometrical, mathematical and computational challenges. I would like to thank Schut Geometrical Metrology and Lute Kamstra for sharing their knowledge and providing me with a practical use case for this research.

My tutors Dr. Henk Bekker and Prof. Dr. Wim H. Hesselink have given me valuable feedback on the contents of this thesis. I'm especially grateful to Dr. Bekker for our almost weekly discussions and input which have helped me immensely throughout the research process.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRO

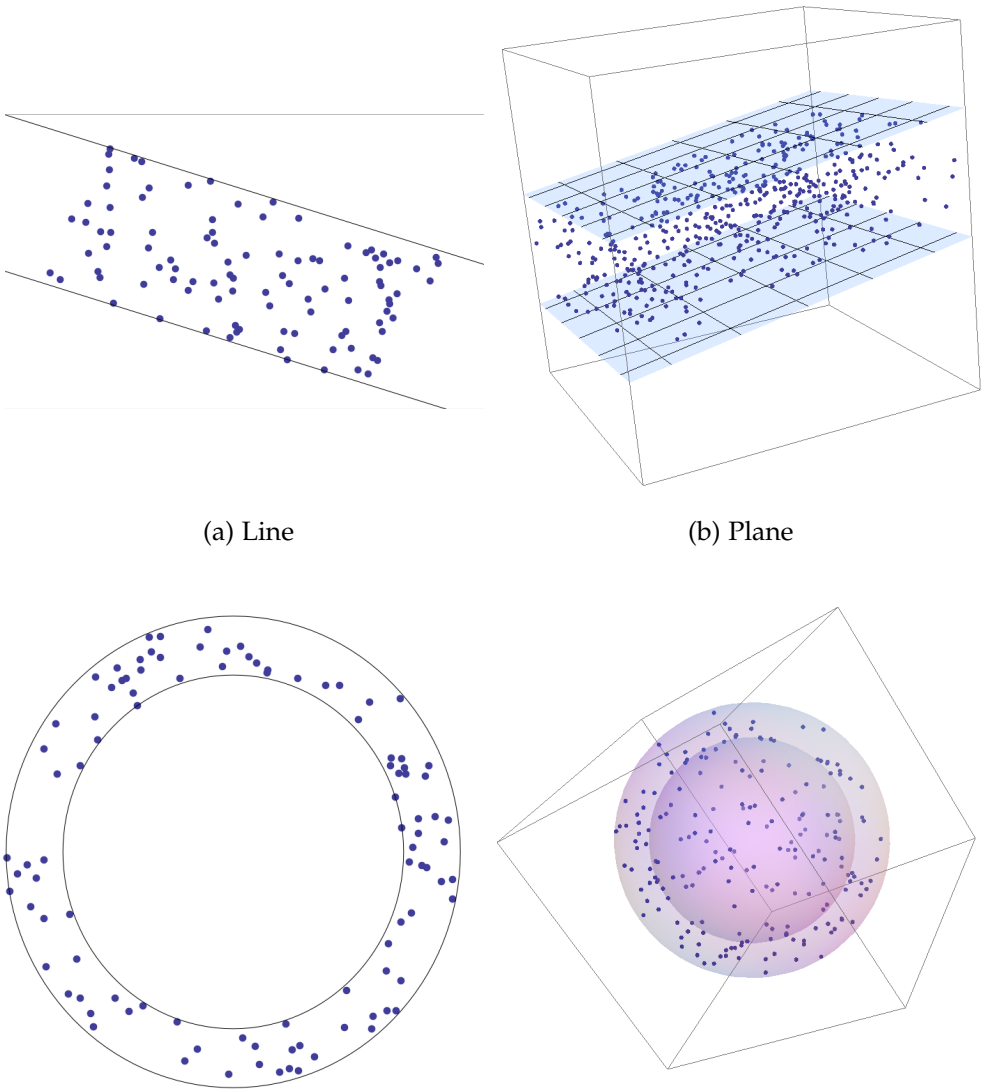
Determining the degree of cylindricity of a point set is a difficult problem. A standard way of determining the cylindricity is by computing the distance of two cylinders fitted tightly from both the inside and outside of the point set. The main use of this error measure is in determining the quality of manufactured cylindrical parts. Coordinate measuring machines can measure a part and return a cloud of measured points which needs to be analyzed to see if they are within production tolerances. The minimal distance between the pair of cylinders can be used to accept or discard a part. Other uses are in reconstruction of cylindrical surfaces and data compression.

This research focuses on two distinct problems. First, we discuss the minimal point set defining both a single cylinder or a pair of coaxial cylinders. We parametrize the cylinder/pair of cylinders in terms of this minimal set of points. Secondly, we compute the cylindricity of a point set P of N points using the parameters obtained by the first part. We present a brute force combinatorial algorithm which always finds the globally minimal cylindricity.

This work is performed in cooperation with the company Schut Geometrical Metrology [1]. Schut manufactures coordinate measuring machines primarily for use in metrology. A cylindrical shell error algorithm is developed to analyze point sets measured by these machines. The algorithm must be implemented in c++ without use of external software programs or libraries.

The cylindricity metric, also called the form error, is defined by the ANSI Standard Y 14.5M. The standard states that the error of a geometrical shape is the smallest distance between two parallel enclosing geometrical elements. For example, for a given set of points, the error of a sphere is computed by calculating the radial distance between the smallest enclosing and the largest inscribing concentric spheres (see fig 1.0.1). Although this may seems like a relatively easy problem, it is nontrivial to compute for complex shapes. In the case of a cylinder shape, the main problem is that no assumption is made on either the cylinder direction or position. This means a cylinder production axis, called the *datum*, may not be used to simplify the problem. In general, the datum is the virtual ideal line, plane, point or axis depending on the form error being evaluated. Without prior knowledge of the datum, actually measuring the error is a complex problem for anything but the most basic geometrical shapes.

1 Schut Geometrical Metrology (http://www.schut.com/Contact/index.htm).



(a) Line (b) Plane

(c) Circle (d) Sphere

Figure 1.0.1: Minimum zone solutions for different geometrical shapes.

Before considering the minimal cylindrical shell for a set points, we focus on a smaller problem: determining the minimal set of points defining cylinder or pair of coaxial cylinders.

Much work has been done on fitting a cylinder through a set of points using nonlinear estimation methods. A drawback of these methods however is that they require a good starting estimate to numerically converge to an approximate solution. Surprisingly little research is done on the parametrization of a cylinder for a minimal point set. Bottema and Veldkamp show that 5 points in general position are needed to restrict the locus of equal distance points to an axis [4]. They also note that given 5 contact points, there are up to 6 real cylinders which pass through these points.

Lichtblau proved that there are either 0, 2, 4 or 6 cylinders through 5 points [20]. In addition he defines the cylinder parameters with respect to 5 instantiated points . Although Lichtblau defines the cylinder parameters in terms of 5 points, it is only usable for instantiated/numerical points. We extend this method to express the cylinder direction and localization in terms of 5 symbolical point coordinates. For this we will solve a system of bivariate polynomial equations. The system of equations is eliminated into a set of univariate polynomials. These are then solved using the CORS [3] common root finding method.

Next we consider the problem of the minimal set of points defining a set of coaxial cylinders, also called a cylinder shell. Hodgson et. al. have shows that 6 points describe a family of cylinder shells. However to our knowledge no research has been done on finding all cylinder shell through these 6 points. We solve this problem for all distributions of 6 points over the inner and outer cylinder. The related systems of equations are again solved using the CORS method.

The second part of this research attempts to compute the minimal cylinder shell through a set of N points, where $N \geq 5$. Computing the minimal cylindrical shell is generally done by improving an initial guess resulting in an approximation of the global minimal cylinder pair. Many different methods exist that approximate the solution. A first approach is gradual improvement of the axis by iterative rotations [6, 8, 13]. Many other simplify the problem by projecting the points onto a plane normal to the estimated axis, in essence reducing the problem to a circle form error [10, 22]. Another common approach solves the nonlinear cylindricity optimization problem by linearizing the problem [8, 9, 12, 23]. The cylindrical axis is even found by Devillers and Preparata using the axis of a hyperboloid as an approximation [10].

Our work attempts to compute the exact global minimal cylinder shell for a point set and thus compute the global minimal cylindricity. The algorithm computes the minimal cylindricity by applying the shell parametrization for 6 points for all combinations of 6 points in the total point set. The method guarantees to find the global solution within numerical precision. The drawback of this approach is the high time complexity of the combinatorial algorithm. The time complexity of the method is $\binom{N}{6} \cdot N$, which makes this method impractical for large point sets.

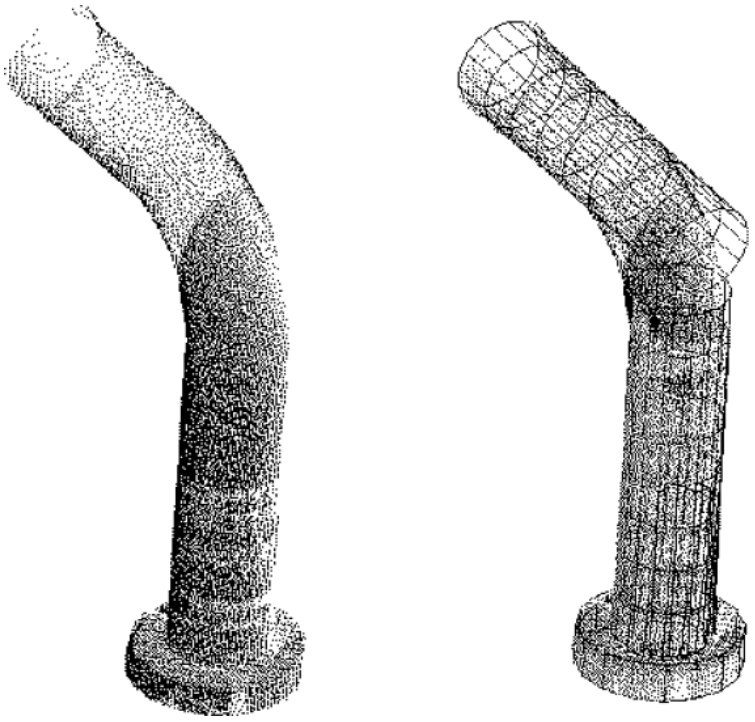
Finally, because of the combinatorial algorithm high time complexity, several heuristic methods are presented. These method use the combinatorial solution to find the minimal cylinder pair but for a smaller point set, generally just 6 points.

Our research extends work on cylinder parametrization, we introduce a novel method of parameterizing a cylinder shell and we present a novel exact cylindricity algorithm. First, we extend the method by Lichtblau for the cylinder parameters in terms of 5 points. A set of univariate polynomi-

als are precomputed for a system of polynomials. This allows the cylinder parameters to be computed quicker than before and without the need for a complex solver at run time. Secondly, we define a pair of coaxial cylinders in terms of the minimal set of points. For this we considering all combinations of 6 points over the two cylinders. Finally we present a method for computing the exact cylindricity of a point set by calculating the minimal shell out of all enclosing cylinder shells. While the method is mentioned by other researchers, to our knowledge, no actual implementation has been investigated.

Many applications for the cylindricity metric and the cylindrical shell computation exist. The minimal shell is used to determine the error of a cylindrical shape for uses in metrology. This is especially useful because the error can be computed with a multipurpose point measuring machine rather than requiring specific measurement tools for just cylindricity. Given a form calculating algorithm the same tool may also be used to compute circularity, flatness and other form tolerances (see figure 1.0.1). Another application for cylindrical shell computation is in surface reconstruction. Here basic shapes are retrieved from a surface scan of a scene. For example, cylinders can be reconstructed from a 3D laser surface scan of some piping (figure 1.0.2). Combining cylinder reconstruction with other geometric shapes reconstruction one can see how an entire scene can be reduced to primitives. A third application is for use in data compression. Spacial patterns in data can be simplified with geometric primitives. These primitives then generalize the data to fewer components and thus compress the data. Finally the exact global cylindricity from our method can be used to as a ground truth for comparison between cylindricity algorithms. Current research uses other investigations to validate their method. Our algorithm allows for comparison with a know truth for any point set small enough to compute.

This work is structured as follows. First we look at the general problem set of finding the minimal pair of shapes around a point set in chapter 2. Next we consider the minimal point set that defines a single cylinder in chapter 3. Chapter 4 considers the minimal point set for a pair of coaxial cylinders. Chapter 5 discusses our combinatorial cylindricity method. Next other heuristic approaches for computing the cylindrical tolerance zone are presented in chapter 6. Results are presented and discussed in chapter 7. Finally, the most important findings along with possible extensions are presented in chapter 8. Additional theory, theorems and supporting concepts are given in the appendix, chapter A.



[7]

Figure 1.0.2: Reconstruction of cylinders from a point cloud.

2

PRELIMINARIES

Before we present our method of finding the minimal margin cylinder pair we discuss the general problem of the minimal distance between two parallel enclosing geometrical shapes.

2.1 FORM ERROR PROBLEM SET

The cylindricity of a point set is just one problem in the broader problem set known as the form error problem set. The form error describes the error of a point set for any geometric shape.

The general form tolerance is the minimum zone between two parallel/-coaxial/concentric shapes. For degenerative shapes (such as circles, spheres and cylinders) two additional tolerances are possible. Both the maximum inscribed and minimum circumscribed circle/sphere/cylinder are used as form tolerances.

Previous work has been done in finding the minimal: planes [8, 14, 16, 17, 19], circles [1, 2, 12, 15, 21], spheres [2] and cones [23]. The form error for many of these shapes can be solved exact and with polynomial time complexity. Only the form error for cylinders and cones is approximated by heuristic methods. The heuristic methods are relatively quick, but do not guarantee the global form error and may instead return a local minimum.

2.2 CYLINDRICITY ERROR

The cylindricity error is a metric for describing the radial distance between a pair of coaxial cylinders fully inscribing and enclosing a point set. The ANSI Standard Y 14.5M defines the cylindricity tolerance as:

> "Cylindricity is a condition of a surface of revolution in which all points of the surface are equidistant from a common axis. A cylindricity tolerance specifies a tolerance zone bounded by two concentric cylinders within which the surface must lie."

Similar metrics exists such as the metric based on only the minimum enclosing or maximum enclosed cylinder. These are defined by another standard ISO/DIS 12180-1 and 12180-1-2 [10]. In the field of metrology however the pair of cylinders defined by the ANSI standard is more commonly used. Although the standard distinguishes cylindricity and cylindricity tolerance, we use both terms to indicate the pair of cylinders fully enclosing a point set.

3 CYLINDER FROM POINTS

Before we consider the complex problem of the minimal cylinder pair for many points, we want to know the minimal point set for a single cylinder. We aim to answer the following questions: how many points define a cylinder? Do these points define a unique cylinder or a family of cylinders? And finally, can we describe our cylinder in terms of its points? Other researchers have already investigated these problem [4, 20]. We take an additional step that defines a cylinder direction and position in terms of the minimal set of points.

Unless stated otherwise, all point sets are considered to be in general position. This entails that no two points are the same, no three points are collinear and no four points are coplanar. This allows us not to be concerned with degenerate cases in the process of discussing our method. Later we will look at some cases where points are not in general position.

Section 3.1 discusses the number and type of points that describe a cylinder. Section 3.2 reformulates the cylinder axis parameters in terms of the input points. Finally, section 3.2.2 covers the procedure for point not in general position.

3.1 MINIMAL NUMBER OF DEFINING POINTS

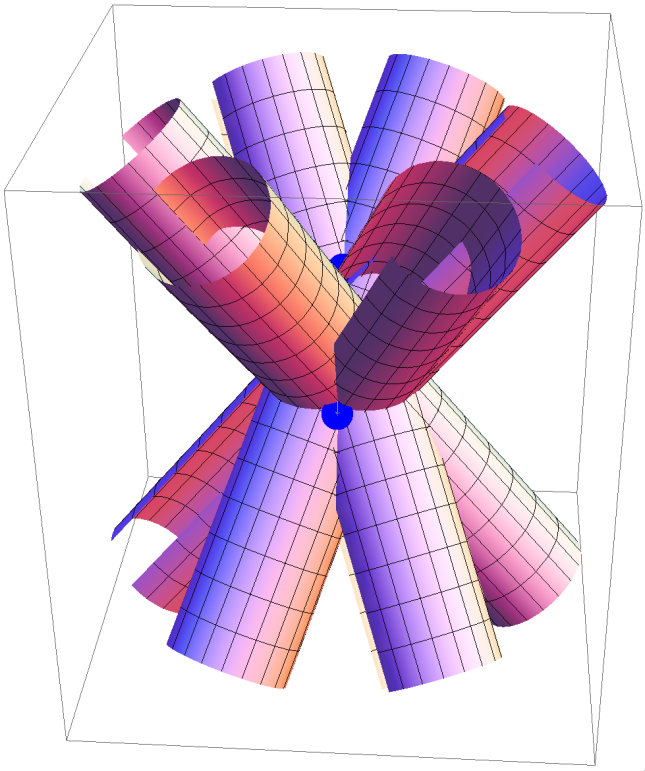
The minimal set of points for basic 3D shapes is quite easy to comprehend, 2 points define a line, 3 points define a plane. It is however harder to imagine the number of points that define more complex 3D shapes. In further discussion of these points, we will call these defining points the shape's surface *contact points*, *feature points* or *spanning points*.

At first we will only consider how many points are needed to define a cylinder or a set of cylinders. This means that initially we do not consider the number of cylinders that fit for the point set. For ease of discussion we therefore mention the problem as finding the cylinder through a set of points, even though there may actual be a family of cylinders that satisfy the problem.

The minimal number of points that define a cylinder can be determined by considering the *degrees of freedom* (DOF). As seen in appendix A.1, a cylinder is defined by 5 parameters. 2 of these 5 parameters describe the axis direction, 2 define the axis position and the last parameter is the radius of the cylinder. With no points to describe the cylinder, all 5 variables are unrestricted giving 5 degrees of freedom. With each feature point the DOF

is reduced by 1. A total of 5 points restrict the DOF to 0, thus defining one or multiple cylinders rather than a continuous set of cylinders.

The formal proof of why a minimum of 5 points is sufficient to define a cylinder is nontrivial. The proof has been presented in 1977 by Bottema and Veldkamp [4]. Plucker coordinates are used to describe a cylinder. Starting with 3 points, the family of possible cylinders is reduced by considering a fourth, fifth and eventually a sixth point. They prove that 5 points define a set of at most 6 real cylinders. These 6 solutions exists for any set of points defined by the vertices of two tetrahedra with a common face. Work by Lichtblau further proves that the number of real solutions is always even, so either 0, 2, 4 or 6 cylinders exist through 5 points [20]. A point set for which there are 6 cylinders through the 5 points is seen in figure 3.1.1.



[20]

Figure 3.1.1: 6 cylinders through set of 5 points. Points lie on vertices of shared face double tetrahedra.

3.2 CYLINDER PARAMETERS FROM POINTS

Knowing that 5 points describe 0, 2, 4 or 6 cylinders, we ask ourselves whether the cylinder parameters can be expressed in terms of point coordinates. This would allow us to quickly compute all cylinders for any 5 points. Note that we want to find all cylinders for a minimum set of points. Later,

when computing the cylindricity, we aim to find the minimal cylinder pair for N points.

In the selection of our parametrization method we have two major considerations. First, we don't want to use any external software libraries and programs. The main reason is that the cylindricity computation is only a minor part of the large software product created by Schut. Such a small component should not introduce additional dependencies. Secondly the method should be optimized for speed. The underlying reason is that the procedure of finding the cylinder and similarly the cylinder shell for a set of points is extensively used by our cylindricity calculation method. Any small improvement in performance for this procedure will greatly improve global performance.

Ideally the parameters of the family of cylinders can be written symbolically in terms of the 5 point coordinates. This would allow constant time calculation of the parameters and will not require any external computational methods. To achieve this we first consider solving the equations for the cylinder variables with use of the Gröbner basis. As it turns out, this approach is impossible in acceptable time. So in lieu of a complete symbolic technique a method which greatly reduces the problem is presented.

There are two separate stages involved in parametrization. Part of the parametrization can be precomputed and another part can only be computed at run time. Expression of cylinder parameters in terms of the point coordinates can be largely precomputed. Computing the parameters for instantiated points can only be done at run time. Clearly we aim to do as many calculations as possible before run time so computation at run time is minimal. As mentioned performing all calculation before program execution using Gröbner basis does not work. So we look into a hybrid method which partially precomputes the problem and solves the rest at run time.

The current method presented by D. Lichtblau directly inserts the instantiated points into a set of cylinder equations and uses a numerical polynomial system solver to obtain the parameters [20]. While this method is ideal for the problem of computing the number of cylinders through 5 points, it is quite slow and requires a bivariate polynomial system solver. In practice such a solver would have to be provided by external software, e.g.: Mathematica or Maple. The dependency on such an external package would break one of our requirements.

We present a method that does a large part of the computation beforehand. So at run time the problem is reduced to the point where a simple univariate polynomial solver suffices.

3.2.1 *Symbolic*

If possible we would like to reformulate a cylinder in terms of the coordinates of the 5 contact points. We will attempt rewrite the cylinder pa-

rameters using the Gröbner basis for various cylinder parametrization approaches.

The Gröbner basis for a set of equations would be easily solvable by a kind of forward substitution at run time (see appendix A.4). We have attempted to compute the Gröbner basis for both the transformation and functional approach cylinder equations (see appendix A.1). In theory for each of these equations a Gröbner basis exists. In practice though, computing the basis took an impractical amount of time (>2 hours) and was likely not computable in a sensible time frame.

Although the Gröbner basis finding algorithm will always find the correct solution, for higher order polynomials with many variables the computation may take very long due to a high time complexity. This is the case for the equations we have attempted to solve. For example, the transformation equations consisting of 4 polynomial equations of degree 4 with 4 free variables. On top of the free variables in these equations, the basis computation is further complicated by the presence of 3 symbolic coordinates constants for each of the 5 points.

The Gröbner basis for a system of equations for all 4 parameters can be computed when the system is simplified by instantiating part of the points coordinates. We were able to compute the Gröbner basis in acceptable time when just 2 coordinates of the 15 coordinates were not instantiated. The other 13 coordinates consisted of random numerical values. Any more symbolic constants and computing the Gröbner basis does not complete within reasonable time (<2 hours).

So if all points are instantiated, the cylinder parameters are solvable using Gröbner basis in acceptable time. We however are interested in rewriting the cylinder equations symbolically which proved impossible in acceptable time. This means the system can not be solved symbolically for the cylinder parameters. We therefore present a different method of which partially rewrites the cylinder equations so computation at run time is simplified.

3.2.2 *Elimination*

As it turns out, completely rewriting point coordinates to cylinder parameters by means of Gröbner basis is infeasible. We therefore resort to a different method which partially simplifies the problem. A polynomial system of equations is manipulated into a set of univariate equations.

We use the bivariate polynomial system introduced by Lichtblau [20]. This bivariate system is eliminated to two univariate polynomials. A characteristic of these univariate polynomials is that they contain all of the roots of the original bivariate system. At run time the univariate polynomials are solved for instantiated points. These roots are paired in order and used to retrieve the cylinder parameters.

Cylinder equation

The cylinder notation suggested by Lichtblau results in a system of two bivariate 3rd order polynomials which can be solved using elimination [20].

Their nifty approach defines a cylinder consisting of two directional parameters and two equations in these direction variables. This is done in the following manner. A point set P is symmetrically preconditioned into set P' (see appendix A.3). The cylinder direction is then represented as a vector

$$D = \begin{pmatrix} a \\ b \\ 1 \end{pmatrix}. \tag{3.2.1}$$

All 5 points in P' are then projected onto an plane through the origin with normal D. Let points $P'' = \{P_0, \ldots, P_4\}$ where P'' is this projected point set. If a cylinder passes through all points in P' then a circle will pass through points P_0,P_1 and P_2. From the circle equations, the squared radius, $rsqr$, and center, C, are determined. The cylinder position is equal to the circle position in 3D

$$C = \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix}. \tag{3.2.2}$$

Now in order for the cylinder direction parameters to be valid, points P_3 and P_4 need to be considered. For a cylinder with direction D at position C the following two constraints also need to be satisfied

$$\begin{cases} d(P_3) & = rsqr, \\ d(P_4) & = rsqr, \end{cases} \tag{3.2.3}$$

Here, d is the squared distance for point p to the axis formed by D and C. Inserting points P_3 and P_4 into equation 3.2.3 gives two bivariate equations in variables a and b

$$
\begin{cases}
f(a,b) = & b^3\left(x_2^2 z_3 - z_3\right) + a^2 b\, y_2^2 z_3 - 2\, b^2 a\, x_2 y_2 z_3 \\
& +a^2\left(y_2 z_3^2 + y_3^2 y_2 - y_2^2 y_3\right) \\
& +b^2\left(-y_3 x_2^2 + x_3^2 y_2 + y_2 z_3^2 - y_2 + y_3\right) \\
& +a\, b\left(2\, x_2 y_2 y_3 - 2\, x_3 y_2 y_3\right) - 2\, a\, x_3 y_2 z_3 \\
& +b\left(x_2^2 z_3 + y_2^2 z_3 - 2\, y_2 y_3 z_3 - z_3\right) \\
& +x_3^2 y_2 - x_2^2 y_3 + y_2 y_3^2 - y_2 - y_2^2 y_3 + y_3, \\
g(a,b) = & b^3\left(x_2^2 z_4 - z_4\right) + a^2 b\, y_2^2 z_4 - 2\, a\, b^2 x_2 y_2 z_4 \\
& +a^2\left(y_2 z_4^2 + y_4^2 y_2 - y_2^2 y_4\right) \\
& +b^2\left(-x_2^2 y_4 + x_4^2 y_2 + y_2 z_4^2 - y_2 + y_4\right) \\
& +a\, b\left(2\, x_2 y_2 y_4 - 2\, x_4 y_2 y_4\right) - 2\, a\, x_4 y_2 z_4 \\
& +b\left(x_2^2 z_4 + y_2^2 z_4 - 2\, y_2 y_4 z_4 - z_4\right) \\
& +x_4^2 y_2 - x_2^2 y_4 + y_2 y_4^2 - y_2 - y_2^2 y_4 + y_4,
\end{cases}
\tag{3.2.4}
$$

where x_i, y_i and z_i are the x, y and z coordinates respectively for point P_i. The solution of this system of bivariate polynomial equations f and g are root pairs (a, b). Such a root pair represent cylinder axis orientation. The other cylinder parameters are all functions of a and b and so can easily be solved from these root pairs. While Lichtblau uses this method to count the number of solutions, we use the notation to describe the cylinder parameters through 5 points.

Because the cylinder direction, D, is always nonzero in the z coordinate, no cylinder can be found if the actual axis cylinder is parallel to the x-y plane (i.e. $a = b = 0$). There is however a quick work around for this problem. Another permutation of points is used to generate the preconditioned point set.

Elimination

The bivariate system of polynomials is reduced to two univariate polynomials using the elimination method (see appendix A.5). The resultants of the elimination process are computed using Mathematica, a symbolic computation program. Direct elimination of the system in equation 3.2.4 is infeasible in reasonable time (< 2 hours). Note however that this system is actually just a general bivariate polynomial of degree 3

$$
\begin{cases}
c_1 b^3 + c_2 a^2 b + c_3 b^2 a + c_4 a^2 + c_5 b^2 + c_6 b\, a + c_7 a + c_8 b + c_9 & = 0 \\
c_{10} b^3 + c_{11} a^2 b + c_{12} b^2 a + c_{13} a^2 + c_{14} b^2 + c_{15} b\, a + c_{16} a + c_{17} b + c_{18} & = 0
\end{cases},
\tag{3.2.5}
$$

where constants c_i are the point coordinates from the specific 3rd order cylinder polynomial. Because the point set P is symmetrically preconditioned equation f and g do not contain a a^3 term. The values for c_i are

$$\begin{aligned}
c_1 &= x_2^2 z_3 - z_3, \\
c_2 &= y_2^2 z_3, \\
c_3 &= -2 x_2 y_2 z_3, \\
c_4 &= -y_2^2 y_3 + y_2 y_3^2 + y_2 z_3^2, \\
c_5 &= -x_2^2 y_3 + x_3^2 y_2 + y_2 z_3^2 - y_2 + y_3, \\
c_6 &= 2 x_2 y_2 y_3 - 2 x_3 y_2 y_3, \\
c_7 &= -2 x_3 y_2 z_3, \\
c_8 &= x_2^2 z_3 + y_2^2 z_3 - 2 y_2 y_3 z_3 - z_3, \\
c_9 &= -x_2^2 y_3 + x_3^2 y_2 - y_2^2 y_3 + y_2 y_3^2 - y_2 + y_3, \\
c_{10} &= x_2^2 z_4 - z_4, \\
c_{11} &= y_2^2 z_4, \\
c_{12} &= -2 x_2 y_2 z_4, \\
c_{13} &= -y_2^2 y_4 + y_2 y_4^2 + y_2 z_4^2, \\
c_{14} &= -x_2^2 y_4 + x_4^2 y_2 + y_2 z_4^2 - y_2 + y_4, \\
c_{15} &= 2 x_2 y_2 y_4 - 2 x_4 y_2 y_4, \\
c_{16} &= -2 x_4 y_2 z_4, \\
c_{17} &= x_2^2 z_4 + y_2^2 z_4 - 2 y_2 y_4 z_4 - z_4, \\
c_{18} &= -x_2^2 y_4 + x_4^2 y_2 - y_2^2 y_4 + y_2 y_4^2 - y_2 + y_4.
\end{aligned} \tag{3.2.6}$$

Variable elimination is applied to the general system of bivariate equations (see appendix A.5). The system from equation 3.2.5 is eliminated into the univariate set of equations

$$\begin{aligned}
F(a) &= d_1 a^8 + d_2 a^7 + d_3 a^6 + d_4 a^5 + d_5 a^4 + d_6 a^3 + d_7 a^2 + d_8 a + d_9 \\
G(b) &= d_{10} b^8 + d_{11} b^7 + d_{12} b^6 + d_{13} b^5 + d_{14} b^4 + d_{15} b^3 + d_{16} b^2 + d_{17} b + d_{18}
\end{aligned}, \tag{3.2.7}$$

where the constants d_j are constant in terms of c, which are dependent only on point coordinates. Notice that while equation 3.2.5 represents a system of equations, equation 3.2.7 is no longer a linked system of equations. This is desired because we want to obtain univariate polynomials rather than a system of bivariate polynomials. But this also entails that the roots are no longer paired, meaning that roots from the univariate polynomials need to be recombined. The process of root pairing is considered at a later stage.

The lower power d's are formed by many terms and the higher power constants are expressed by fewer terms. Because the equations of the d

constants are very large, full listing would be impractical. For example, the smallest constant d_1 in variable a is

$$\begin{aligned} d_1 \;=\; & c_{10}^2 c_{13} c_2^3 + c_1 c_{12}^2 c_{13} c_2^2 - 2 c_1 c_{10} c_{11} c_{13} c_2^2 - c_3 c_{10} c_{12} c_{13} c_2^2 \\ & + 2 c_1 c_4 c_{10} c_{11}^2 c_2 - c_1 c_4 c_{11} c_{12}^2 c_2 + c_3 c_4 c_{10} c_{11} c_{12} c_2 + c_1^2 c_{11}^2 c_{13} c_2 \\ & + c_3^2 c_{10} c_{11} c_{13} c_2 - c_1 c_3 c_{11} c_{12} c_{13} c_2 - c_1^2 c_4 c_{11}^3 - c_3^2 c_4 c_{10} c_{11}^2 \\ & - c_2^2 c_4 c_{10}^2 c_{11} + c_1 c_3 c_4 c_{11}^2 c_{12} \,. \end{aligned} \tag{3.2.8}$$

The constant for the same power b term is smaller. This is because the bivariate polynomial system is simplified to not having a a^3 term. The eliminated univariate polynomial in b is simpler than the univariate polynomial in a. For example, the highest power in b has coefficient

$$d_{10} = -c_{10} c_{11} c_3^2 + c_2 c_{10} c_{12} c_3 + c_1 c_{11} c_{12} c_3 - c_2^2 c_{10}^2 - c_1^2 c_{11}^2 - c_1 c_2 c_{12}^2 + 2 c_1 c_2 c_{10} c_{11}. \tag{3.2.9}$$

This coefficient is less than half in "size" than the same power coefficient in a. For other coefficients this difference is even more pronounced. For example, where the polynomial in b could "fit" on half an a4 paper, the equations for variable a would cover over 4 pages.

Root solving.

The univariate polynomials expressed in input point coordinates obtained in the previous section can be solved by a univariate root solver.

Root solving requires the points to be numeric rather than symbolic. Because numerical values are only known at run time, all the following computations are performed at run time.

In the previous section the cylinder equations were reformulated to a pair of univariate polynomials of degree 8. These eight degree polynomials have at most 8 roots. In practice it turns out that these univariate roots are actually of highest order 6. Both polynomials are solved using a Laguerre root solver (see appendix A.6). This yields up to 6 roots in a and b, which is to be expected knowing that Bottema and Veldkamp have shown that there are at most 6 real cylinders [4].

Common roots

The elimination step produces two disjoint sets of univariate equations. The resulting roots found using the Laguerre root solver are thus disjoint. We use the original bivariate equations to pair the roots again.

We use the Combinatorial Optimisation Root Selection (CORS) method [3]. This method works as follows. Consider a graph $H = (V, E, w)$ with vertices $V = V_1 \cup V_2$, where V_1 and V_2 are the root sets u_i and v_j for univariate function F and G respectively. The set of edges E is defined by $(i, j) \in E \subseteq V_1 \times V_2$.

Each edge weight is calculated by evaluating the univariate roots in the original bivariate polynomial system 3.2.4. The original bivariate equations will evaluate to zero when substituting a valid root solution pair. The edge weights are defined in a weight matrix w

$$w_{i,j=}\sqrt{\left|f(u_i,\, v_j)^2\right| + \left|g(u_i,\, v_j)^2\right|}. \tag{3.2.10}$$

The graph H is known as a bipartite graph. A feasible matching π is a permutation which maps V_1 onto V_2. The total graph weight for a permutation π is: $w(\pi) = \sum_{(i,j)\in\pi} w(i,j)$. We are concerned with finding the minimal over all permutations. This is known as the *Linear Assignment Problem* (LAP).

The linear assignment problem can be solved in $O(k^3)$, for example by using the Hungarian algorithm. We however take a shortcut and best case solve the problem in $O(k^2)$. The main reason for this is not the reduced time complexity, but rather not requiring a complex linear assignment solver. The method assumes that the best a and b root pairs are the minimal cells in the weight matrix row and column. This allows us to simply loop over all root pairs and pick the minimal one from either the row or column.

The result of this step are the root pairs of the original system of bivariate polynomial equations.

Root to parameters

Recall that a root pair of equation 3.2.4 represents the direction of the cylinder axis. From these directions the center and radius are calculated. But all these parameters are based on a preconditioned point set P'. To get the cylinder parameters for the original points P, we apply the inverse preconditioning transformation. The scalar value is simply scaled, the direction is rotated and the center are scaled, translated and rotated.

The complete process from input points to cylinder parameters is illustrated in figure 3.2.1.

Exception case

Up till this point we have considered all points P to be in general position. There are however many degenerate cases imaginable. In practice we have only run into a single situation. Consider the case were 3 points in P are on a line. After preconditioning the first 3 points in P' are still on a line. The used cylinder equation formulation assumes there is a cylinder direction for which there is a circle through the projected points P_0, P_1 and P_2. But clearly, no matter how the points are projected, no circle can pass through 3 points on a line. Therefore this case is degenerate. We handle this exception by using a different permutation of the 5 input points. As long as not all

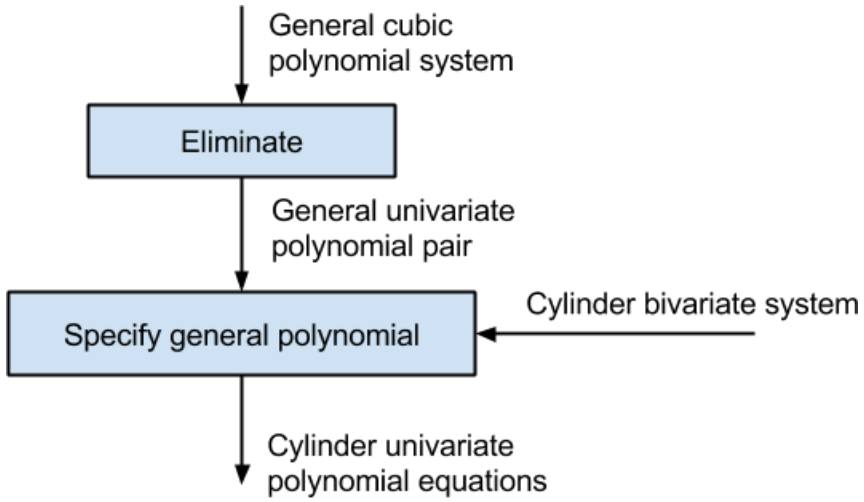
points are collinear there is a permutation for which a circle passes through points P_0, P_1 and P_2.

Not a general method

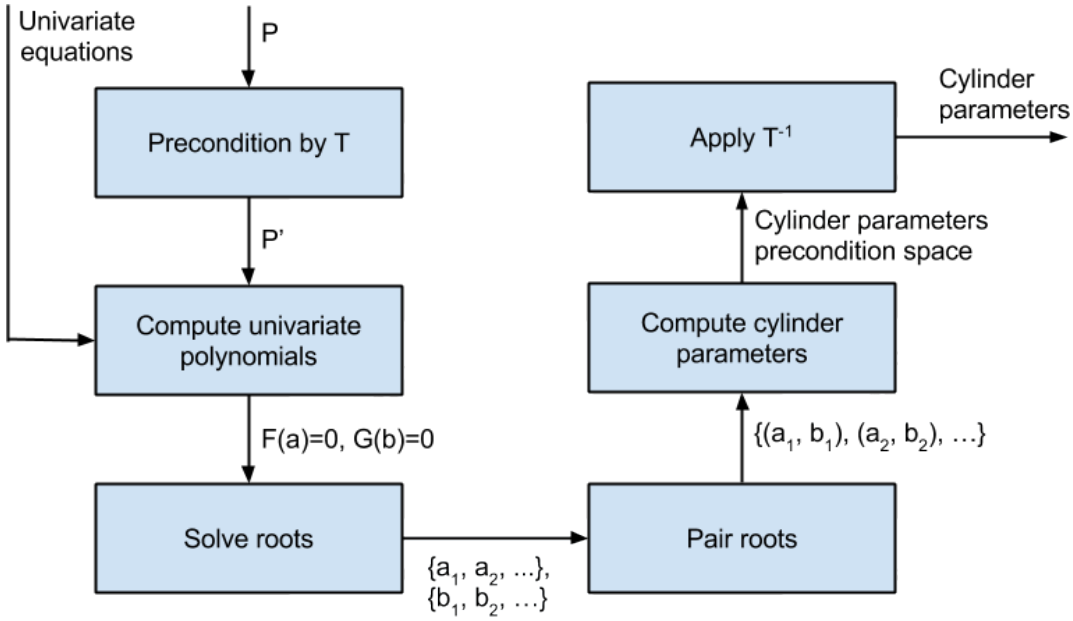
It is noteworthy that the usage of the CORS technique rather than another bivariate system solver comes at the cost of some generality. The computed univariate polynomial set is specific to this and only this problem of finding a cylinder through 5 points. For any other bivariate system a new set of univariate polynomials need to be calculated. As we will see such is the case in the next chapter, where we determine a cylinder shell through a set of 6 points.

3.3 SUMMARY

A set of up to 6 cylinders is defined by 5 points. Fully expressing the cylinder set parameters in terms of point coordinates proved infeasible within acceptable computation time. Another method which partially precomputes the cylinder system is presented. With use of elimination and common root solving the cylinder parameters are computable in terms of the 5 input points.



(a) Precomputation of the univariate root equations.



(b) Run time procedure. Points set P is preconditioned, Next the univariate equations from the precomputation step are instantiated. The univariate roots are computed and paired. The cylinder parameters in preconditioned space are computed and finally transformed to input space.

Figure 3.2.1: Process of computing the cylinder parameters for a point set P.

4

CYLINDER SHELL

Having considered the problem of a single cylinder through 5 points, we now focus on the points which define the family of cylinder shells. Again we first determine the number of points required to fix the cylinder shell in space. Next we rewrite the input coordinates cylinder shell parameters. For the latter we distinguish 3 situations depending on how the feature points are distributed over the cylinder pair.

4.1 CONTACT POINTS

The points defining a cylinder shell can lie either on the inscribed or circumscribed cylinder. Hodgson et. al. show that 6 points describe a cylinder shell [13]. There are 5 different distributions of points over the inner and outer cylinder

- $1 \ll 5$,
- $2 \ll 4$,
- $3 \ll 3$,
- $4 \ll 2$,
- $5 \ll 1$,

where the notation $a \ll b$, means there are a points on the inscribed and b points on the circumscribed cylinder. Knowing that 5 points define a cylinder it should be clear that just one more point is required to fix the radius of the second cylinder. This means a total of 6 points describe the set of cylinder shells.

4.2 SHELL PARAMETERS FROM POINTS

We want to determine cylinder shells for 6 points, $shells_6(P)$. The shell equations are even more complex than those for a single cylinder, so Gröbner basis computation again would not work.

Like the cylinder calculation, we again use the projected circle approach and solve the common roots using the CORS method. Because the point distribution varies over the two cylinders we distinguish several cases. Each of these cases has a different system of bivariate polynomials and therefore requires solving a different set of univariate polynomials.

4.2.1 *Elimination*

Again the cylinder shell parameters are computed using the CORS method. We use the projected circle approach to find the cylinder shell

$$CS = \{d_x, d_y, d_z, p_x, p_y, p_z, r_1, r_2\}, \tag{4.2.1}$$

consisting of two cylinders A and B. Roots a and b of direction equation 3.2.1 are used so

$$\begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} = \begin{pmatrix} a \\ b \\ 1 \end{pmatrix}. \tag{4.2.2}$$

p_x, p_x and p_y are the coordinates of a point on the cylinder axis fixing it in space, they equal c_x, c_y and c_z from equation 3.2.2. Parameters r_1 and r_2 are the radii for cylinders A and B.

Notice that only the computation of the univariate polynomial set varies with respect to the cylinder calculation. This means that only the preprocess procedures are redone (figure 3.2.1a). Computing the actual parameters from the univariate polynomials remains the same (figure 3.2.1b).

First we will look at the distribution of the points over the inner and outer cylinder of the cylinder shell.

Subproblems

As discussed, a cylinder shell has 6 contact points, P, which are distributed over both the inscribed and circumscribed cylinder. The 5 possible combinations are $1 \ll 5$, $2 \ll 4$, $3 \ll 3$, $4 \ll 2$, $5 \ll 1$. In our mathematical approach of the problem we are however not concerned whether a point lies on the inner or outer cylinder, but rather just whether it lies on cylinder A or B. We denote these combinations as $a \sim b$ where a and b are the number of points that lie on cylinder A and B respectively. Not distinguishing inner and outer points, the 5 distributions are reduced to just three different cases

- $5 \sim 1 = 1 \sim 5 = 1 \ll 5 \cup 5 \ll 1$,
- $4 \sim 2 = 2 \sim 4 = 2 \ll 4 \cup 4 \ll 2$,
- $3 \sim 3 = 3 \ll 3$.

The shells for 6 points are the same as the union of shells for each of the three cases

$$shells_6(P) = shells_{5\sim1}(P) \cup shells_{4\sim2}(P) \cup shells_{3\sim3}(P). \tag{4.2.3}$$

Each of these cases has a selection of points for cylinder A and B. This means all combinations for these cases need to be visited. For example, for

the $5 \sim 1$ case, any of the 6 points can lie on the second cylinder, so 6 combinations are possible. The order of the points is unimportant when selecting the subset $A \in P$ and $B \in P \setminus A$. This means we are dealing with a true combinatorial problem, not a permutation. The number of combinations for each of the three cases can be calculated using the binomial coefficient

$$C(5 \sim 1) = \binom{6}{5} = 6, \tag{4.2.4}$$

$$C(4 \sim 2) = \binom{6}{4} = 15, \tag{4.2.5}$$

$$C(3 \sim 3) = \binom{6}{3} = 20, \tag{4.2.6}$$

where $C(case)$ is the number of combinations for that *case*. The point indices for the three cases are shown in table 1. Notice that for the $3 \sim 3$ case the set is repeated halfway. So the real number of combinations for $3 \sim 3$ is actually $\frac{1}{2} C(3 \sim 3) = 10$. This means to check all cases a total of $6 + 15 + 10 = 31$ point combinations are considered. We will consider the computation of shell for each case separately.

Cylinder shells for the $5 \sim 1$ *case*

The $5 \sim 1$ case is solved by computing the shells for all combinations of 5 points out of 6. The cylinders through 5 points is calculated using the method presented in section 3.2.2. This yields the parameters d_x, d_y, d_z, p_x, p_y, p_z and r_1 for the cylinder shell. The second radius, r_2 is the distance of point 6 to the shell axis.

Cylinder shells for the $4 \sim 2$ *and* $3 \sim 3$ *cases*

For the $4 \sim 2$ and $3 \sim 3$ cases new equations are derived. Like the cylinder, we describe the shell in terms of two axis direction parameters and two equations. Again the cylinder direction is $D = \{a, b, 1\}$. All points P are symmetrically preconditioned to obtain set P' (see appendix A.3). All points in P' are projected onto the orthogonal plane to D yielding points P'', where $\{P_0..P_5\} \in P''$. The center and radius are defined by the circle through points P_0, P_1 and P_2. The equations depend on whether the $4 \sim 2$ or $3 \sim 3$ case is considered.

In case of the $4 \sim 2$ situation, the remaining point P_4 is on cylinder A which is of radius r_1. For a feasible solution, the other two points on B are the same distance to the cylinder axis. Together this gives the constraints

$$\begin{cases} d(P_3) = r_1^2 \\ d(P_4) = d(P_5) \end{cases}, \tag{4.2.7}$$

where d is the squared distance for a point to the axis defined by D, p_x, p_y and p_z. This bivariate polynomial system is again a cubic polynomial pair in variables a and b. The monomials are all terms of a and b up to degree 3 except a^3.

In the symmetrical case of $3 \sim 3$, the equations are all based on points on B. All these points are equal distance to the cylinder axis

$$\begin{cases} d(P_3) = d(P_4) \\ d(P_4) = d(P_5) \end{cases}. \tag{4.2.8}$$

Both systems are eliminated to a pair of univariate equations. These equations are then used at run time by the CORS method to compute the roots for the univariate equations seen in equation 4.2.7 and 4.2.8.

Parameters from roots

Analogously to the cylinder case the, root pairs define the shell direction. Again the axis position is calculated from the direction and the radii are computed by point line distance. The computed shell is initially calculated for the preconditioned point set P'. By the inverse transformation of the preconditioning we obtain the cylinder shells in the original input space.

Ordered cylinders

Although cylinder shells are now described in terms of unordered cylinders, it is worth understanding the consequences of distinguishing an inner and outer cylinder.

If we would use a constraints such that $r_1 < r_2$, then we would have a system of three constraints in two variables. For example the $4 \ll 2$ case would be

$$\begin{cases} d(P_3) = r_1^2 \\ d(P_4) = d(P_5) \\ r_1^2 < r_2^2 \end{cases}. \tag{4.2.9}$$

These constraints are a lot more complex to solve because of the addition of an inequality constraint. A small attempt has been made to eliminate this system, but computation using a symbolic solver failed (timeout).

Being able to distinguish between inner and outer points can be beneficial to performance as we briefly discuss in the final chapter 8.

4.3 SUMMARY

6 points are needed to define a discrete set of cylinder shells. The 6 points can be divided in 5 ways over the inner and outer surface. By reformulating this to combinations over surface A and surface B, only 3 different combinations are possible. For each of these combinations the cylinder shells set is expressed in terms of input points by using elimination and common root finding. Knowing how to write a shell in terms of points allows us to consider the problem of the minimal shell over many points (next chapter). These concepts also aid in the more heuristic methods of finding the cylindricity, chapter 6.

Cylinder A	Cylinder B
0	1,2,3,4,5
1	0,2,3,4,5
2	0,1,3,4,5
3	0,1,2,4,5
4	0,1,2,3,5
5	0,1,2,3,4

(a) Set $5 \sim 1$.

Cylinder A	Cylinder B
0,1	2,3,4,5
0,2	1,3,4,5
0,3	1,2,4,5
0,4	1,2,3,5
0,5	1,2,3,4
1,2	0,3,4,5
1,3	0,2,4,5
1,4	0,2,3,5
1,5	0,2,3,4
2,3	0,1,4,5
2,4	0,1,3,5
2,5	0,1,3,4
3,4	0,1,2,5
3,5	0,1,2,4
4,5	0,1,2,3

(b) Set $4 \sim 2$.

Cylinder A	Cylinder B
0,1,2	3,4,5
0,1,3	2,4,5
0,1,4	2,3,5
0,1,5	2,3,4
0,2,3	1,4,5
0,2,4	1,3,5
0,2,5	1,3,4
0,3,4	1,2,5
0,3,5	1,2,4
0,4,5	1,2,3
——	——
1,2,3	0,4,5
1,2,4	0,3,5
1,2,5	0,3,4
1,3,4	0,2,5
1,3,5	0,2,4
1,4,5	0,2,3
2,3,4	0,1,5
2,3,5	0,1,4
2,4,5	0,1,3
3,4,5	0,1,2

(c) Set $3 \sim 3$.

Table 1: Shell point combinations, notice that the combinations for $3 \sim 3$ repeat half way.

5

COMBINATORIAL MINIMAL ZONE

In this chapter we answer the main question of this investigation, that is, how to compute the global cylindricity for points set P. Up till here we have considered the problem of finding all cylinders and cylinder shells for minimal number of points. Here on the other hand we want the single minimal cylinder shell for N points.

Being able to describe a cylinder shell based on 6 points, we can finally consider the broader problem of the minimal shell for N points. The minimal shell is computed from all shells for 6 points out of N.

This method will guarantee the global minimal solution at the cost of significant time complexity. Nevertheless it is a useful technique for small point sets and provides a way to get the actual cylindricity of a point set as ground truth for other methods.

First we present the algorithm. Next we present the time complexity. And finally we look at a parallel form of the algorithm to partially mitigate the poor time complexity.

5.1 ALGORITHM

The minimal shell of N points is the same as the minimal shell for all combinations of 6 points of N. Let Q be all combinations of 6 points for point set P. The set of all possible shells is

$$S = \bigcup_{q \in Q} shells_6(q). \tag{5.1.1}$$

A shell needs to enclose the entire point set P in order to be considered a valid coaxial pair. A shell s encloses the set P if

$$valid(s) = \forall_{p \in P}\, s.r_1 \leq d(p, s) \leq s.r_2, \tag{5.1.2}$$

where $d(p, s)$ is the distance from point p to the cylinder axis of shell s. Symbols $s.r_1$ and $s.r_2$ represent radii r_1 and r_2 for cylinder shell s where $r_1 < r_2$. The set of all valid shells is

$$V = \{s \in S | valid(s)\}. \tag{5.1.3}$$

The minimal cylinder shell is simply smallest cylindrical shell in V

$$minimalShell(V) = \min_{v \in V}(v.r_2 - v.r_1). \tag{5.1.4}$$

The pseudo code for this method is shown in algorithm 5.1. By considering all combinations, by definition the algorithm is guaranteed to find the global cylindricity minimum.

Algorithm 5.1 Brute force

```
while q is not last combination P:
  q=nextCombination(P)
  shells = shells_6(q)
  for shell s in shells:
    for p in P \ q:
      if s.r_1 ≤d(p, shell) ≤ s.r_2 and cylindricity(s) < cylindricity(best)
        best = shell
      endif
    endfor
  endfor
endwhile
return best
```

5.2 EXCEPTION CASES

The presented combinatorial algorithm works for point sets of size $N \geq 6$. In case $N = 5$, no discrete set of shells through those 5 points exists. There is however a minimal set of shells which have an equal radius inner and outer cylinder. Such a same radius cylinder shell is in essence just a single cylinder. All of these shells have no spacing between the cylinders and are thus of cylindricity 0.

For $N \leq 4$ there is no longer a discrete set of possible cylinders. An infinite number of cylinders touch $N \leq 4$ points. A subset of these infinite cylinders through N points has cylindricity 0 and is thus minimal.

So, an infinite number of cylinder shells exists for which the cylindricity is 0 when $N \leq 4$. If $N = 5$, there is a discrete set of minimal cylinder shells of cylindricity 0.

5.3 COMPLEXITY

This combinatorial algorithm has quite poor performance for everything but the smallest point sets. The number of combinations of 6 out of N points is $\binom{N}{6}$. For each pair of 6 points, 31 different sub combination are tried. These 31 sub combinations in turn define multiple cylinder shells. For each of these shells, N points need to be tested whether they are inscribed and circumscribed by the cylinder shell. This leads to computation time of $\binom{N}{6} \cdot 31 \cdot N \cdot c$, where c is some unknown constant. The resulting combinatorial time complexity is

$$\theta_{combinatorial}(N) = \begin{pmatrix} N \\ 6 \end{pmatrix} \cdot N = \frac{N!}{6!(N-6)!} \cdot N = N^6 \cdot N = N^7. \quad (5.3.1)$$

A graph for the time complexity with respect to input size is seen in figure 5.3.1.

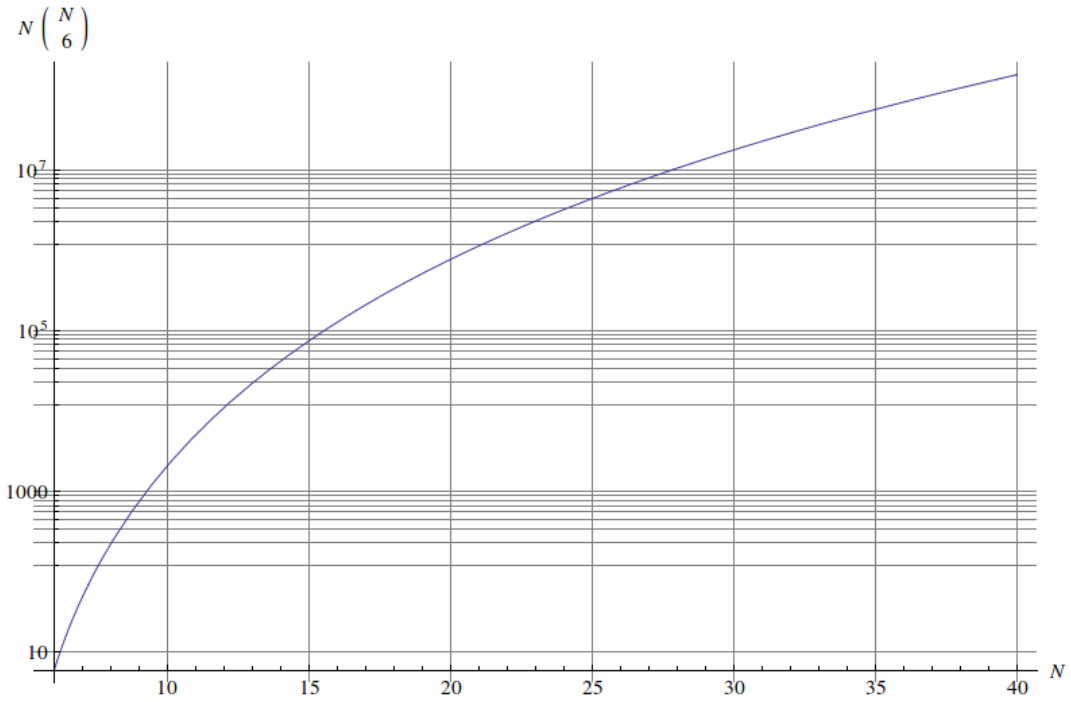


Figure 5.3.1: Combinatorial time complexity graph for point set of size N.

N	$\binom{N}{6} \cdot N$
10	2100
15	75075
20	775200
25	4427500

Table 2: Samples combinatorial cylindricity time complexity.

5.4 PARALLELIZATION

The brute force algorithm is an ideal case for parallelization. Any of the combinatorial loops are subject for naive parallelization. The most logical loop to perform in parallel is the outer loop, all combinations of 6 points. Each thread can do calculations on a subset of the combinations. The minimal shell out of the set of minimal shells can be determined at the end of

the loop, requiring no locking/communications overhead during minimal computation for each combinations subset. We have tested a parallel version of this algorithm where the outer loop is parallized using the OpenMP for loop directive.

5.5 SUMMARY

By trying all combinations of points the global solution to the minimal zone problem is guaranteed to be found. This method is both simple to implement and not susceptible to local minima. The major drawback is of course the time complexity of $\binom{N}{6} \cdot N$. The number of combinations quickly grows to large for many practical uses. The algorithm is however highly parallelizable, partially mitigating the high time complexity.

6 HEURISTIC METHODS

As seen in the previous section, computing the best exact solution to the cylindricity problem is infeasible for all but the smallest point sets. Here we look at methods which compute approximate solutions by making some assumptions on the solution to achieve better performance. All these heuristic methods here are very experimental. The main focus of this investigation is on computing combinatorial exact solution (chapter 5), rather than these approximate solutions. The algorithms presented here should be considered as possible alternatives, trading precision for speed.

Before looking at different heuristic methods, we briefly look at the least square solution. This method is occasionally used as a cylindricity measure, but in general cases it is far from the minimal zone cylindricity. We use the least square line for a point set as a cylinder start axis for the various heuristic method. Because of this, it plays a crucial role in convergences to either a local or the global minimum.

The steepest descent method by Cheraghi et. al. will be discussed [8]. This method iteratively improves the cylinder normal by trying out small rotations and storing the best. For each cylinder normal its center is determined by first projecting the points onto the plane with the same normal through the origin. For these projected points the smallest width annulus is computed.

The second method is a combinatorial approach, but uses a subset of all points to have faster convergence. Each iteration, the exact minimal shell is computed for all combination of 6 points in this smaller subset. A selection of points for the next iteration is generated after each iteration.

Finally a genetic algorithm is presented. Individuals representing sets of 6 spanning points are evolved until a local minimal cylindricity is found. This method resembles the random search pattern of simulated annealing, but allows discrete sets to be used.

6.1 INITIAL AXIS

Almost all heuristic methods require a good initial guess to converge to the global minimum rather than a local minimum. Often as start situation the least squares solution for the point set is computed.

The least square solution for a point set can refer to two different things. An extension to the least square method used in regression from 2D to 3D results in a plane with minimal average distance to the points. We however are interested in finding a line with minimal average distance to the points.

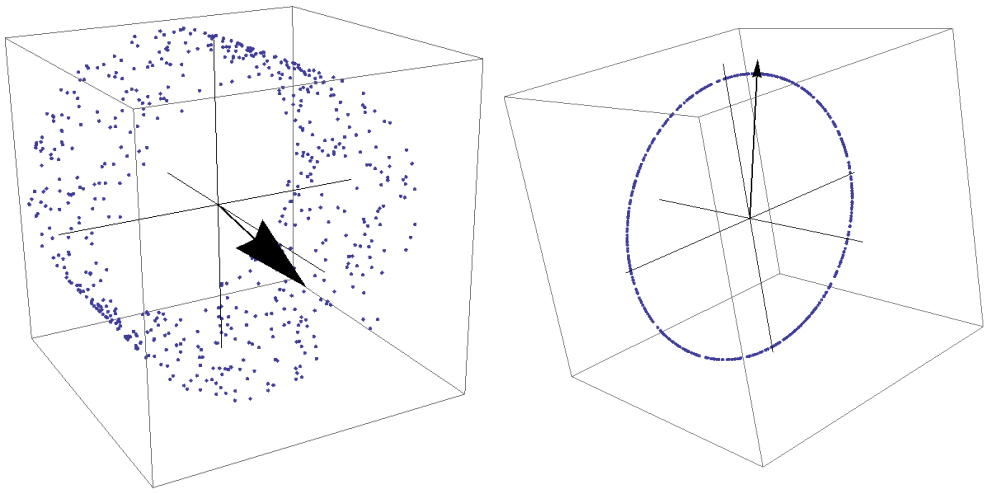
We therefore use the least square method based on principal component analysis (see appendix A.2).

L.W. Foster has found from extensive experimentation that

> "The Least Squares solution lies close to the minimum zone solution [11], and thus provides a good initial solution.".

The least square solution may lie close to the minimal zone axis for typical cylindricity problem cases. There are however many cases imaginable where the least square solution is completely off from the global solution. Many of these are purely theoretical. Figure 6.1.1 shows a problem which may well be encountered in a practical situation. The problematic point set has a large radius and a relatively small elongation along the axis. Because of this, the first principal axis is almost orthogonal to the actual cylinder axis.

We will use the least square solution as start situation for our methods because in general it is accurate, fast and simple to implement. The point sets used are quite cylindrical so the least square line is close to the cylinder axis, see figure 6.1.1a.



(a) Cylindrical point set, least square is close to actual cylinder axis.

(b) Very narrow cylindrical point set, least square is orthogonal to actual cylinder axis.

Figure 6.1.1: Least square initial cylinder for some sets.

6.2 STEEPEST DESCENT

The first heuristic method we discuss iteratively improves the cylinder by probing nearby solutions. The method was proposed by Cheraghi, Hossein and Jiang [8].

The method uses a sort of steepest descent approach. Initially the least square line is used as a starting cylinder. Next the axis is rotated in order to obtain the steepest descent. However, rather than using the partial derivatives for this step, samples are taken by rotating the axis in 8 different directions. These 8 rotation axes are perpendicular to the cylinder axis and are evenly distributed, i.e. make an angle of $k\frac{2\pi}{8}$, where $k \in \{0, .., 7\}$. A rotation μ is applied around each axis.

For each rotated axis direction the points are projected onto a plane orthogonal to the axis. For the projected 2D point set the minimum width annulus is computed. The cylindricity of this minimal annulus is the same as the cylindricity of the cylinder in 3D space. The cylinder position is computed from the minimal width annulus center by converting the 2D coordinates of the plane back to 3D coordinates.

At the end of each iteration the best cylinder from the 8 rotations is stored and the step size is halved μ. The algorithm finished when a desired step size has been achieved.

6.3 SUBSET COMBINATORIAL

The second method investigated reduces the combinations problem of a large point set to an iterative combinations problem of only a few points. The assumption is that, given a cylinder axis, a better solution likely exists by combining only the first few outer and inner points with respect to the current axis. Our method is a variation on the algorithm by Hodgson et. al. [13].

Initially a cylinder is constructed from the least square line solution. Next all points are sorted on distance to the cylinder axis. A point set Q is made from the closest and furthest k number of points. All possible cylinder shells, CS, are calculated from this subset Q. At the end of each iteration the best cylinder is the minimal combinatorial cylinder of all shells CS. The algorithm continues until the new cylindricity is equal or larger than the previous cylindricity. The algorithm may terminate before finding the local minimum.

Hodgson et. al., loop over the shell point distribution cases, $1 \ll 5$, $2 \ll 4$, $3 \ll 3$, $4 \ll 1$ and $5 \ll 1$, and pick l inner and m outer points depending on the distribution. Our method can not make the distinction between inner and outer points. In order to cover all point distributions we therefore have to select k inner and k outer points such that all combinations $1 \sim 5$, $2 \sim 4$ and $3 \sim 3$ can be constructed from this subset. In order to support the $1 \sim 5$ case, we require $k \geq 5$. Having k inner and outer points means that each such subset is of at least $2k \geq 10$ points.

6.4 GENETIC ALGORITHM

Finally, we have created a genetic algorithm that uses the exact cylinder shell parametrization. Individuals representing subsets of 6 points from the entire point set are iteratively evolved. Their genes are passed on through selective breeding, creating populations with increasingly better cylindricity.

This method iteratively improves a population of individuals by mutation, selection and breeding. The order in which this is done is given in algorithm 6.1. First a random population G_0 of size M is created from all N points in set P. Each iteration for a random subset of individuals a mutation occurs with probability ϕ. The fitness of the population is calculated by computing the minimal cylindricity for each individual. A new generation of individuals is then bred from the fittest individuals. This new generation G_{i+1} consists of individuals with genetic material that is a combination of parent material.

Algorithm 6.1 Combinatorial genetic algorithm.

G_0 = random_population(M)
i=0
error_estimate = infinity
while error_estimate > λ
  G_i = mutate(G_i) with probability ϕ
  set_fitness(G_i)
  S = select(ρ, μ, G_i)
  G_{i+1} = breed(S)
  error_estimate = best_fitness(G_{i+1}) - best_fitness(G_i)
  $i = i + 1$
endwhile
cylindricity = lowest fitness G_i

Each individual has a strand of genetic material representing a subset of 6 points from point set P. An individual I consists of the subset

$$I = \cup_{\{1..6\}}(j)\,,\ j \in \{0..N-1\}\,. \tag{6.4.1}$$

The initial population G_0 consists of M random individuals. An individual is created by generating 6 integers between $[0, N)$, taking care not to introduce duplicate integers.

The genetic material of an individual in a population is mutated with a probability ϕ. If an individual I is selected for mutation, a random feature point is exchanged for a point with index r, where $r \in [0, N) \setminus I$.

The cylindricity of each individual I determines their fitness. This cylindricity is calculated using the combinatorial minimal cylinder shell for the feature points of I.

The new generation, G_{i+1} is created by combining individuals from the current generation G_i. Individuals are selected for breeding based on their fitness value. The fittest ρ number of individuals are selected for breeding and μ random individuals are selected from the remaining set. Random unfit samples are included in order to keep genetic variation in following generations. We call the selected set of individuals S.

M new individuals are generated for generation G_{i+1}. Individuals in generation G_{i+1} are created by cross over from random parents I_a and I_b in S. A new individual I_{new} represents a subset of point for which k points are from parent a and $6-k$ number of points are from parent b. The value for k is a random value between 1 and 5. I_{new} is

$$I_{new} \quad = \quad I_a[0:k] \cup I_b[k:6]. \tag{6.4.2}$$

I_{new} may contain duplicate points from this cross over step. If I_{new} contains duplicate points we simply pick two other random parents from S and repeat the process until a child is created without duplicate points.

All these procedures combined simulate the evolution of a set of individuals with cylindricity as their fitness value. We repeat the computation of new generations until either a maximum number of iterations is reached or the minimal cylindricity between subsequent generations is below a value λ.

Lai et al, have also presented a genetic algorithm for cylindricity calculation [18]. The key difference in their genetic algorithm is that their individuals represent cylinder parameters whereas our individuals represent point subsets. This means their method searches a continuous parameter space whereas the combinatorial genetic algorithm searches only the discrete set of combinations of P.

6.5 SUMMARY

We have discussed how the least square solution can be used as a starting point for heuristic algorithms. Three different heuristic methods are presented: steepest descent, subset combinatorial and a genetic combinatorial algorithm.

7 RESULTS EN DISCUSSION

Having discussed all methods we will present our results and compare them with previous work. First we will discuss cylinder and cylinder shell parametrization results. Next we discuss the combinatorial cylindricity with respect to previous research. Next we present the running times of the minimal zone cylindricity method for varying input size. Finally we discuss the cylindricity results for the heuristic methods presented.

Results have been produced using a i7-3610QM CPU at 3.4GHz. This CPU has 4 hardware cores and uses hyper threading to simulate 8 logical processors.

7.1 5 POINT CYLINDER

Computation of a cylinder through 5 points is at the core of many complex operations. In a programming sense, the procedure is often nested within a double, triple or even quadruple loop, and thus greatly influences the total running time. On average the computation of all cylinders through 5 points takes 40ms on system 1.

Occasionally cylinder solutions through 5 points are missed because of numerical precision. The process of solving the bivariate system of equations is quite susceptible to numerical errors. Especially the evaluation of univariate root pairs in the original bivariate system. Valid pairs occasionally produce very high weights even though the result should be close to 0. The linear assignment algorithm is therefore unable to properly pair univariate roots causing some solutions to be missed. These situation occur most frequently when the cylinder axis is almost on the x-y plane, resulting in large roots.

These exceptional cases could potentially be detected and a different permutation of points could be used to circumvent this problem. In our implementation however no steps are made to handle this exception.

7.2 6 POINT CYLINDER SHELL

5 distributions of points over the cylinder pairs through 6 points exist. We look how the points are distributed over the cylinder pairs for many cases. Next we discuss how many cylinder shells are defined by 6 feature points.

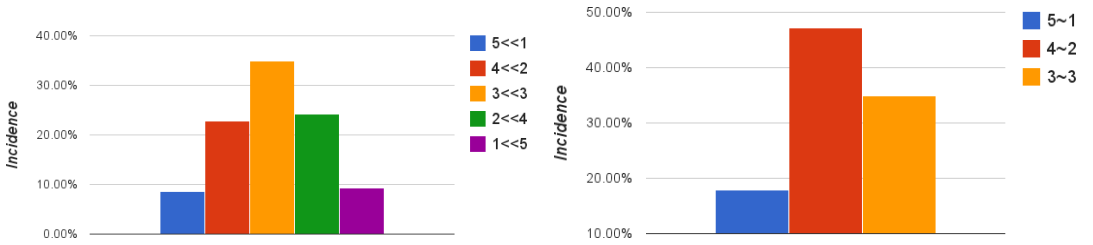
Distribution

The distributions of points over the two cylinders is investigated for 100 random sets of 6 points. The distribution of the feature points over the two cylinders for both order dependent and invariant is shown in figure 7.2.1 and table 3. The order invariant results are obtained by combining the incidence results for the $1 \ll 5$, $5 \ll 1$ and $2 \ll 4$, $4 \ll 2$ cases.

Note that there are an equal number of cases of inner and outer point distributions. For example, the $2 \ll 4$ and $4 \ll 2$ case both have an incidence of about 23%. In most cases the 6 points are distributed evenly over the two cylinders.

The difference in incidence of the different cases can fully be explained by the number of combinations considered. If we take into account the number of combinations for $5 \sim 1$, $4 \sim 2$ and $3 \sim 3$ cases, all distributions occur equally often.

If the incidence of a certain distribution would be very low we could consider skipping it in our combinatorial algorithm for sake of performance. However all distributions occur relatively frequently, so we can not ignore any distribution without losing a significant portion of the search space.



(a) Inner-outer point distribution.

(b) Point distribution over two cylinders, order invariant.

Figure 7.2.1: Point distribution over cylinder pairs.

Distribution	Incidence
$1 \ll 5$	8.57%
$2 \ll 4$	22.86%
$3 \ll 3$	35.00%
$4 \ll 2$	24.29%
$5 \ll 1$	9.29%

(a) Order dependent.

Distribution	Incidence
$5 \sim 1$	17.86%
$4 \sim 2$	47.14%
$3 \sim 3$	35.00%

(b) Order invariant.

Table 3: Point distribution incidence.

Number of solutions

The total number of cylinder shells through 6 points is equal to the sum of the number of shells for the three sub cases.

There are 6 point combinations for the $5 \sim 1$ case. Each of these combinations defines a maximum of 6 cylinder shells, the same number of solutions as that of a cylinder through 5 points. Thus a total of $\binom{6}{5} * 6 = 36$ possible cylinder shells for the $5 \sim 1$ case are possible through 6 points.

We don't have a formal proof of the number of solutions for the $4 \sim 2$ and $3 \sim 3$ cases. However through experimentation we have found a maximum of 76 and 50 distinct cylinder shells for $4 \sim 2$ and $3 \sim 3$ respectively. In total, considering all cases, we have found a maximum of 150 cylinder shells.

Intuitively we expect that there are a maximum 6 shells possible for a combination in either the $4 \sim 2$ or $3 \sim 3$ case. Under this assumption there would be an upper limit to the number of shells of

$$\begin{aligned} |shells_6(P)| &= |shells_{5\sim1}(P)| + |shells_{4\sim2}(P)| \\ &\quad + |shells_{3\sim3}(P)|, && (7.2.1) \\ &= 36 + C(4 \sim 2) * 6 + \frac{1}{2} C(3 \sim 3) * 6, && (7.2.2) \\ &= 36 + 90 + 60 = 186. && (7.2.3) \end{aligned}$$

This value is slightly higher than the empirically found maximum of 150 shells. This means our hypothesis of a maximum of 6 shells per combination still holds. The maximum number of possible shells through 6 points is then 186.

7.3 CYLINDRICITY

Table 4 shows the cylindricity obtained by our combinatorial method for a standard set of points. These point sets are often used in other research, the actual sets are given in appendix A.7.

The cylindricity of our method is on par or better than previous results for set 2 and set 4. The combinatorial cylindricity has more significant digits than previously published work. This may very well be because the other researchers have chosen not to publish such detailed numbers. These extra digits are quite crucial though. We have found that around the global minimum many solutions exist with almost the same cylindricity. To distinguish these cases these last few digits play a crucial role.

The major drawback of the combinatorial method is also apparent from the table. Point sets 1 and 4 are 40 points each, making computation of the global minimal shell infeasible. The number of combinations from these 40 points is almost 30 times that of the next largest point set, set 4 with 24 points.

These two observations illustrate both the strength and weakness of our approach. The combinatorial method computes the global cylindricity in great detail but it is limited in use to smaller point sets than other methods.

	Combinatorial	Steepest descent [8]	Linearization [6]
Set 1	-	0.00100	0.001000
Set 2	0.183957331728777	0.18395	0.18396
Set 3	-	0.00942	0.00941
Set 4	0.00278832434212	0.002788	-

Table 4: Combinatorial cylindricity results with respect to previous research. Point sets 1 and 3 are to large to be computed practical time by our combinatorial method. Carr and Ferreira have not presented results for set 4.

Running time

The running time of our method for varying input size is depicted in figure 7.3.1 and table 5. A key observation is the running time results indeed match the theoretical time complexity of N^7 (compare graphs of figure 5.3.1 and 7.3.1). As input size N increases, more combinations are considered and so the running time increases proportionally.

Figure 7.3.2 and table5 show the running time of the combinatorial method for varying number of threads. Doubling the number of processors significantly reduces the running time of the algorithm. For $N = 20$, a 2 threads speedup of $546/283 \approx 1.93$ is obtained. This is an efficiency of $1.93/2 \approx 0.96$ which is almost ideal speedup. For higher number of threads the parallel efficiency decreases. For 4 and 8 threads the speedup is about 3.11 and 4.81 respectively yielding a parallel efficiency of 0.78 and 0.60. The parallel efficiency is lower than we would expect, given that almost no locking is required, but it is still significant. The parallelizability of the combinatorial algorithm means that given enough processors, much larger point sets can be computed in acceptable time.

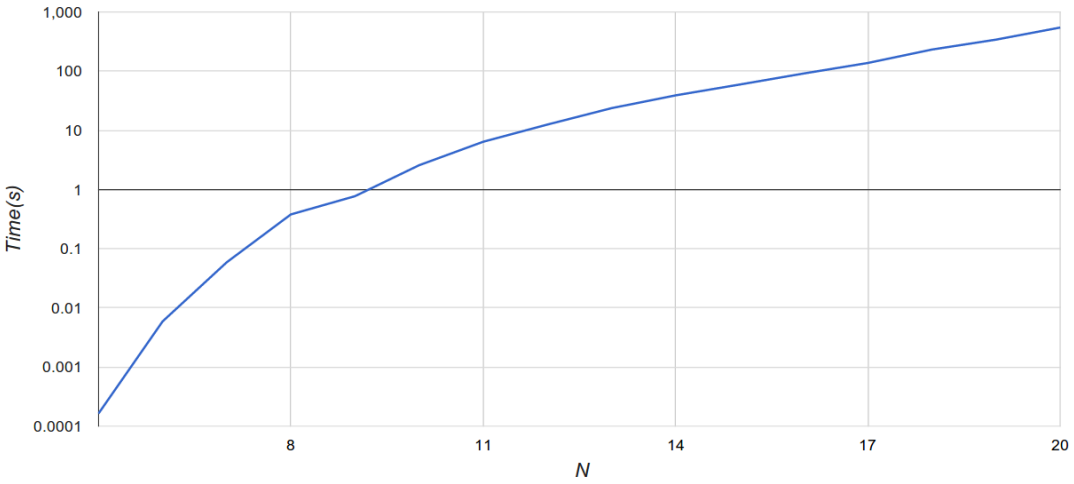


Figure 7.3.1: Combinatorial algorithm running time for different size input with logarithmic time scale.

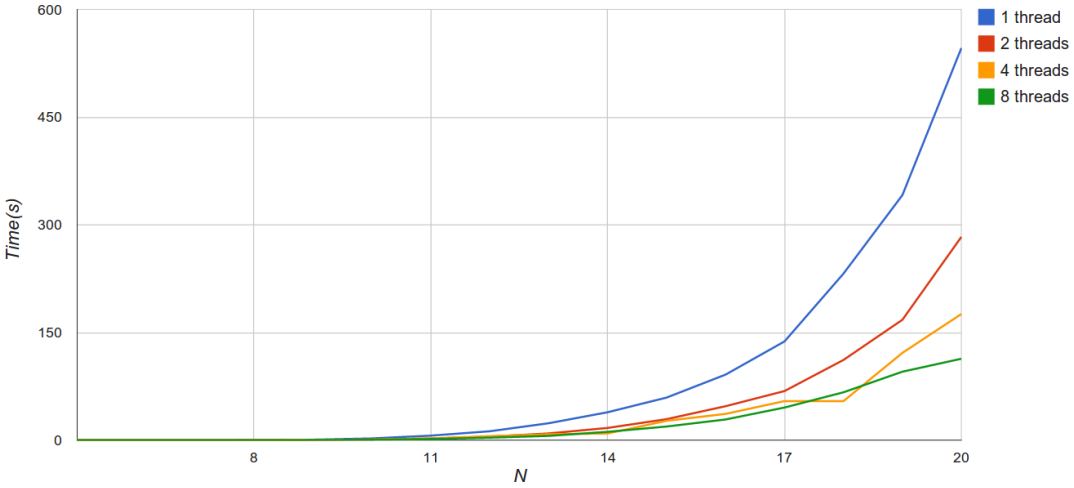


Figure 7.3.2: Parallel combinatorial algorithm running time for varying point set size N for 1, 2, 4 and 8 threads.

N	Single threaded	2 threads	4 threads	8 threads
5	0.0002	0.0002	0.0002	0.0002
6	0.0059	0.0059	0.0059	0.0259
7	0.059	0.0185	0.0591	0.0202
8	0.38	0.08	0.16	0.09
9	0.77	0.24	0.4	0.3
10	2.57	1.1	1.09	0.81
11	6.42	2.42	2.58	1.72
12	12.51	5.08	5.63	3.58
13	23.68	9.51	8.37	6.07
14	39	17.13	9.42	11.73
15	59.43	29.38	27.09	19.11
16	91.4	47.41	36.68	28.92
17	137.64	68.51	54.54	45.55
18	232.06	111.82	54.45	66.78
19	341.47	167.8	121.69	95.46
20	546.23	283.23	175.81	113.5

Table 5: Running time in seconds of both serial and parallel version of combinatorial algorithm for different size inputs.

7.4 HEURISTIC

Finally we discuss the heuristic methods from chapter 6. The cylindricity values and running times of these experimental methods is seen in table 6.

The steepest descent methods converges to the correct cylindricity for sets 2 and 3. In case of set 1 and 4, a local minimum is found rather than the global solution. The method has a low running time with respect to our combinatorial method, but it is quite slow for a heuristic approach. The sub procedure of finding the smallest annulus is responsible for these high running times. This computational and memory expensive smallest annulus procedure is computed 8 times for each iteration. This original algorithm by Charaghi et. al. uses their own smallest annulus solver [8] which does not suffer from poor performance.

The subset combinatorial method iteratively uses the exact calculations from our combinatorial approach to compute the minimal cylinder pair. Because only subsets are considered it is significantly faster than the exhaustive search of the combinatorial method. Unfortunately though, the method needs a very good initial situation in order to converge to the global solution. When the initial axis is close to the global solution, the global cylindricity is quickly found. But the implemented least square initial axis often does not converge to the global solution. Therefore we don't consider this method to work well in practice.

The cylindricity of point set 1 using the genetic algorithm is not found. This is because no single individual in the starting population has a combination of points which describe a cylinder shell enclosing the entire point set. Without valid individuals the algorithm terminates early and no cylindricity value is found. Generally about 10% of the cylinder shells described by the initial population fully encloses the point set. The rest of the individuals do not represent cylinders shells that enclose the point set, and thus have an fitness value of infinity.

The results for the genetic algorithm show that the global cylindricity is often not found. The problem is that an individual representing one or more minimal cylindricity feature points should be considered very fit, but the fitness value doesn't often reflect this. The presence of these minimal cylindricity points, or global feature points, in an individual's point set often does not express itself as a a low cylindricity value. Because of this, selection of the fittest does not work effectively. New generations often have a lower fitness than the previous generation causing early termination of the algorithm.

This discrepancy between minimal cylindricity of a subset of points and its actual fitness values illustrates a important issue with the discrete approach to the cylindricity problem. The problem is that a subset of "good" or even global feature points is not recognizable as such. The cylindricity of a "good" subset may be higher (worse) than that of a "bad" subset of

points. Only combinations of 6 good or global feature points can be recognized as having a low cylindricity. Because of this property, heuristic methods using the cylinder shell parametrization on point subsets have bad or unpredictable results.

		Steepest descent	Subset combinatorial	Genetic algorithm	Known cylindricity
Set 1	Cylindricity	**0.00380701**	**24.8346**	-	**0.001**
	Running time	18.9s	1.20s	-	
Set 2	Cylindricity	**0.184062**	**0.268563**	**0.246036037**	**0.18396**
	Running time	7.46s	13.2s	3.52s	
Set 3	Cylindricity	**0.00996079**	**0.00941013**	**0.061014120**	**0.00942**
	Running time	12.8s	29.2s	3.51s	
Set 4	Cylindricity	**0.00309663**	**0.009111**	**0.005351058**	**0.002788**
	Running time	6.26s	2.48s	4.73s	

Table 6: Cylindricity heuristic methods. No cylindricity value is obtained using the genetic algorithm for point set 1. The known cylindricity values are either from the combinatorial cylindricity algorithm or other publications.

8

CONCLUSION

We have presented methods for computing cylinders and cylinder pairs through a minimal set of points. A new method of computing the cylinders through 5 points using the CORS method is presented. Part of the problem is precomputed making run time computations quicker and requiring only a univariate root solving method.

Secondly a new method of computing cylinder shells parameters for 6 contact points is presented. The solutions are found by considering three distributions of 6 points over 2 cylinders. Of the distributions, the $4 \sim 2$ case is found in nearly half of the solutions, followed by $3 \sim 3$ and $5 \sim 1$ case.

The combinatorial minimal cylinder shell method is presented which uses the cylinder shell parametrization. The method is guaranteed to find the global minimum solution within numerical precision. It however does have a poor time complexity making the method infeasible for large point sets. The algorithm is highly parallelizable, partially mitigating the low performance.

The combinatorial method cylindricity values are on par if not slightly more accurate than previously published results. The global cylindricity value found by our method can be used as a ground truth for further research.

Finally we have presented several heuristic methods using the combinatorial and cylinder shell parametrization. None of them performed very well with respect to either cylindricity value or running time. The main problem is that the cylindricity of a set of 6 points does not well reflect whether a subset of these 6 points is actually good. A near optimum set of 6 points may have a high cylindricity even though a subset of these points in conjunction with some other points would actually have a very low cylindricity.

Both the global cylindricity computation method and cylinder shell parametrization methods could be improved in various ways. A key improvement to the cylinder and cylinder shell solver would be the usage of a more accurate bivariate equation solver. For example, Bekker et. al. suggest using Broydens method for a multiple rooted even degree system such as ours [3].

The time complexity of the combinatorial cylindricity algorithm can likely be improved by distinguishing interior and exterior points. Using the convex hull the outer points of a cylinder shell can be identified. The generation of point combinations would no longer require the selection of 6 points from a total of N points, but rather a points from the smaller set of the con-

vex hull points and b point from from the interior point set. Application of the convex hull however requires the cylinder shell equations to depend on interior or exterior points, whereas at the moment they are interior/exterior invariant. For example when considering the $5 \sim 1$ case only a single exterior point can be used even though in about half of the cases 5 points lie on the exterior cylinder.

Another improvement can be done by further parallelizing the combinatorial minimal shell solver. As shown, the algorithm is very parallelizable and many of the operations are vector operations. This makes this algorithm ideal to be computed by a graphics processing unit. The increase in performance from a highly parallized combinatorial algorithm can increase the practical input size limit. For example, a speedup of a 100 times that of a serial program, using a GPU or another many cores processing unit, would allow an input size of 23 points instead of 12 in equal computation time.

Finally the cylinder shell parametrization for 6 points can be used in heuristic methods for faster convergence. Any heuristic based algorithm will likely converging to 6 feature points. By computing the shell for these points directly, some iterations may skipped.

A

APPENDIX

A.1 CYLINDER EQUATION

Several different mathematical notations describing a cylinder in terms of a set of parameters are commonly used in the literature. Each of these approaches equates a 5 point cylinder in terms of several parameters.

Transformation

One method of writing the cylinder through 5 points is finding the transformation of a point set which equally places all the points at equal distance to a fixed cylindrical axis. Let the z-axis be the cylindrical axis. We need to find the combination of a translation and rotation which transforms the points around the z-axis. The transformation of the point set is

$$Q = R * T\,, \tag{A.1.1}$$

where R and T are homogeneous matrices and matrix Q is the matrix product of R and T . Applying transformation Q to a point x first translates x after which it is rotated. In order to compose transformations into a single matrix all vectors and matrices are augmented/homogeneous vectors and matrices. However, for ease of discussion we will present these transformations using normal matrices and vectors rather than full homogeneous form. The translation transform T, is based on the displacement vector

$$T = \begin{pmatrix} t_x \\ t_y \\ 0 \end{pmatrix}. \tag{A.1.2}$$

Matrix R is chosen such that no periodicity is present in terms of the parameters. The Rodrigues matrix is used because it is defined using quadratic terms rather than trigonometric functions. Another benefit is that the parameters of the Rodrigues matrix correspond to the unit axis rotations. The homogeneous form of the Rodrigues matrix is

$$\frac{1}{1+u^2+v^2+w^2}\begin{pmatrix} 1+u^2-v^2-w^2 & 2(uv-w) & 2(uw+v) \\ 2(uv+w) & 1-u^2+v^2-w^2 & 2(vw-u) \\ 2(uw-v) & 2(vw+u) & 1-u^2-v^2+w^2 \end{pmatrix}. \tag{A.1.3}$$

The parameter w in the Rodrigues matrix reflects a rotation about the z axis. However the distance of points to the z-axis aligned cylinder does not change when they are rotated around the z-axis. w only introduces an unneeded degree of freedom. It is therefore removed (set to 0). The scaling factor before the matrix is removed because it cancels out in a later step. The simplified rotation matrix is

$$R = \begin{pmatrix} 1+u^2-v^2 & 2uv & 2v \\ 2uv & 1-u^2+v^2 & 2u \\ 2v & 2u & 1-u^2-v^2 \end{pmatrix}. \tag{A.1.4}$$

The squared distance of a point, p, to the cylindrical axis is finally computed as

$$d(p) = (Q * p)_x^2 + (Q * pt)_z^2. \tag{A.1.5}$$

The cylinder passing through the 5 points has equal distance to all points. The resulting system of equations is

$$\begin{aligned} d(P_1) &= d(P_2), \\ d(P_1) &= d(P_3), \\ d(P_1) &= d(P_4), \\ d(P_1) &= d(P_5), \end{aligned} \tag{A.1.6}$$

in 4 parameters u, v, t_x, t_y.

The squared norm combined with the squares in the rotation matrix leads to fourth degree equations. For example the distance formula for p_1 after transformation to the z-axis is

$$\begin{aligned} d(P_1) = {} & t_x^2 + t_y^2 + 2t_x^2u^2 - 2t_y^2u^2 + t_x^2u^4 + t_y^2u^4 \\ & +8t_xt_yuv - 2t_x^2v^2 + 2t_y^2v^2 + 2t_x^2u^2v^2 \\ & +2t_y^2u^2v^2 + t_x^2v^4 + t_y^2v^4 \,. \end{aligned} \tag{A.1.7}$$

A big benefit from this method is that the point axis distance is only dependent on the x and y point coordinates after the transformation. A downside however is that the resulting polynomial has many higher order (> 2) terms making them difficult to solve.

Vector notation

A second definition describes a cylinder in terms of a point in space and a direction vector. The orientation T is

$$T = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}. \tag{A.1.8}$$

The cylinder is position in space by vector L

$$L = \begin{pmatrix} L_x \\ L_y \\ L_z \end{pmatrix}. \tag{A.1.9}$$

The 5 point cylinder is again defined as the set of points with equal distance to the axis defined by T and L. The line-point distance computation introduces a lot of complexity because it requires a cross product.

This notation is suited for describing the cylindrical shell problem as a optimization problem. Finding an inscribed and circumscribed cylinder pair for a point cloud is a nonlinear optimization problem [6]. Let r be the radial distance from the axis to the median cylinder of the zone and n be the number of points, then the optimization problem is

$$\min_{i \in \{1..n\}} \max \left| \left\| \hat{T} \times (P_i - L) \right\| - r \right|, \tag{A.1.10}$$

where $\hat{T} = \frac{T}{|T|}$.

Function notation

A third method of defining a cylinder is by use of functions [20]. This method requires no complex transformations or cross products but does require the axis to not be in the x-y plane. As will be shown this however is only a minor problem. For general positioned points the functional cylinder notation is

$$y = ax + b, \tag{A.1.11}$$

$$z = cx + d, \tag{A.1.12}$$

where a and c determine the cylinder direction and b and d are the base of the cylinder. The direction vector, D, and position vector, B, are

$$D = \begin{pmatrix} 1 \\ a \\ c \end{pmatrix}, \tag{A.1.13}$$

$$B = \begin{pmatrix} 0 \\ b \\ d \end{pmatrix}. \tag{A.1.14}$$

All points are again equal distance, r, to the cylinder axis. The distance of a point to the line is computed using the cross product as in the previous section. Again, because only the distance is important we can remove the square in the distance computation. For all preconditioned points this leads to the set of equations

$$
\begin{aligned}
0 &= a^2d^2 - 2abcd + b^2c^2 + b^2 + d^2 - r^2, \qquad \text{(A.1.15)}\\
0 &= a^2 + 2ab + b^2 + c^2 + b^2c^2 + 2cd - 2abcd + d^2 + a^2d^2 - r^2,\\
0 &= a^2d^2 - 2a^2d\mathrm{p4}_z + a^2\mathrm{p4}_x^2 - 2abcd + 2abc\mathrm{p4}_z + 2ab\mathrm{p4}_x\\
&\quad +2acd\mathrm{p4}_y - 2a\mathrm{p4}_x\mathrm{p4}_y + b^2c^2 + b^2 - 2bc^2\mathrm{p4}_y - 2b\mathrm{p4}_y\\
&\quad +c^2\mathrm{p4}_x^2 + c^2\mathrm{p4}_y^2 + 2cd\mathrm{p4}_x - 2c\mathrm{p4}_x\mathrm{p4}_z + d^2 - 2d\mathrm{p4}_z\\
&\quad +\mathrm{p4}_y^2 - r^2,\\
0 &= b^2 + b^2c^2 - 2abcd + d^2 + a^2d^2 - r^2 + 2ab\mathrm{p4}_x + 2cd\mathrm{p4}_x\\
&\quad +a^2\mathrm{p4}_x^2 + c^2\mathrm{p4}_x^2 - 2b\mathrm{p4}_y - 2bc^2\mathrm{p4}_y + 2acd\mathrm{p4}_y\\
&\quad -2a\mathrm{p4}_x\mathrm{p4}_y + \mathrm{p4}_y^2 + c^2\mathrm{p4}_y^2 + 2abc\mathrm{p4}_z - 2d\mathrm{p4}_z\\
&\quad -2a^2d\mathrm{p4}_z - 2c\mathrm{p4}_x\mathrm{p4}_z,\\
0 &= b^2 + b^2c^2 - 2abcd + d^2 + a^2d^2 - r^2 + 2ab\mathrm{p5}_x + 2cd\mathrm{p5}_x\\
&\quad +a^2\mathrm{p5}_x^2 + c^2\mathrm{p5}_x^2 - 2b\mathrm{p5}_y - 2bc^2\mathrm{p5}_y + 2acd\mathrm{p5}_y\\
&\quad -2a\mathrm{p5}_x\mathrm{p5}_y + \mathrm{p5}_y^2 + c^2\mathrm{p5}_y^2 + 2abc\mathrm{p5}_z - 2d\mathrm{p5}_z\\
&\quad -2a^2d\mathrm{p5}_z - 2c\mathrm{p5}_x\mathrm{p5}_z.
\end{aligned}
$$

By using functions to define the cylinder the solution fails for a cylinder which actual axis is on the y-z plane. To circumvent this problem we can do two things without loss of generality. First we can pre-rotate and post-rotate our point set. And secondly we can take any other permutation of the point set.

A.2 LEAST SQUARE

The least square line of a point set is often used as a initial axis for heuristic cylindricity computation methods. The least square line method refers to the use of the first principal component rather than the least square interpolation which is used in regression.

The first principal components of a point set is the eigenvector belonging to the largest eigenvalue. It represent a line through the points centroid and minimizes the sum of squared errors.

The first principal components is calculated as follows. First all points are centered around the mean of the point set in order for the principal components analysis (PCA) to reflect the points variance. Let P' be the mean centered set,

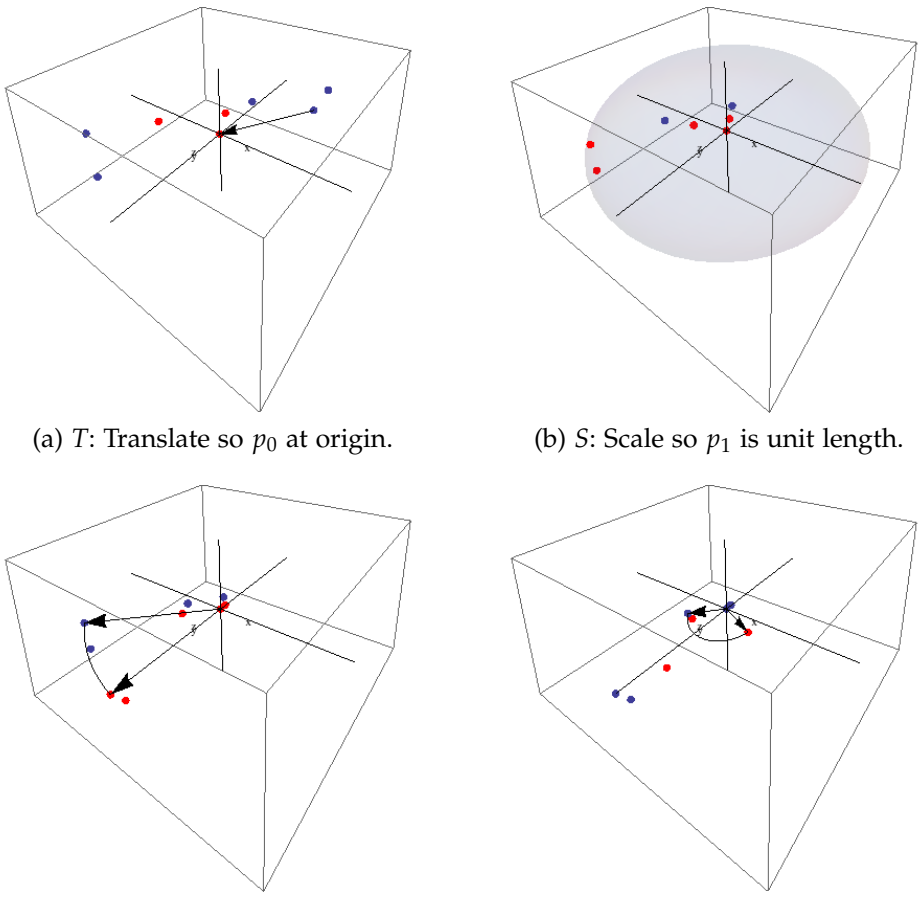
Next the covariance matrix, CM, for point set P' is calculated. Diagonal entries of this matrix are the variances of each points. Off diagonal elements of the matrix represent covariance, a measure of similar direction between different points.

The principal components of P are the eigenvectors of matrix CM. The least square line direction is the first eigenvector of the covariance matrix. Its position is equal to the centroid of point set P.

A.3 PRECONDITIONING

Every point set can be transformed to a simplified point set without loss of generality. For every set of points in $\mathbb{R}^3$ a set of transformations is possible to a point set containing at least six zero coordinates and a coordinate with value 1. We will show a method which accomplishes this through the use of rigid transformations.

Let $\{P_0, ..., P_{N-1}\}$ be the points in set P of size N. The transformation is based on just 3 points in P. First the entire set P is translated by transformation T so that point p_0 is positioned at the origin. Next the set is uniformly scaled by transformation S such that the vector to p_1 has unit length. Finally a set of rotations is performed so that p_1 is on the x-axis and p_2 is in the z plane. This is done by rotation transformations R_1 and R_2 respectively. The entire transformation $Q = R_2 R_1 S\, T$ is illustrated in figure A.3.1.



(a) T: Translate so p_0 at origin.

(b) S: Scale so p_1 is unit length.

(c) R_1: rotate so p_1 is on y-z plane.

(d) R_2: So p_2 is on z-axis.

Figure A.3.1: Transformation to accomplish precondition.

The entire transformation transforms the original point set P

$$p_0 = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}, \; p_1 = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}, \; p_2 = \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix}, \; \ldots, \; p_N = \begin{pmatrix} x_N \\ y_N \\ z_N \end{pmatrix}, \tag{A.3.1}$$

into the simplified set

$$P'_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \; P'_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \; P'_2 = \begin{pmatrix} \sim \\ \sim \\ 0 \end{pmatrix}, \; \ldots, \; P'_N = Q\,P_N\,, \tag{A.3.2}$$

where the symbols $\sim$ can have any value. The introduction of the 6 zeros and a 1 greatly simplifies algorithms and equations based on this preconditioned set.

SYMMETRICAL PRECONDITIONING

The general precondition method results in the point set seen in equation A.3.2. For our calculations an additional operation is performed so the first two points are symmetrical around about y-z plane. An additional scaling, S_2 and translation, T_2, transformation is applied to the point set. That results in the symmetrical transform $Q' = T_2 S_2 R_2 R_1 S\,T$, where

$$S_2 = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{A.3.3}$$

$$T_2 = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}. \tag{A.3.4}$$

The resulting symmetrically preconditioned point set is

$$P''_0 = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \; P''_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \; P''_2 = \begin{pmatrix} \sim \\ \sim \\ 0 \end{pmatrix}, \; \ldots, \; P''_N = Q' P_N. \tag{A.3.5}$$

GENERALITY

Preconditioning of the points set for use in cylinder parametrization can be done without loss of generality. All transformation are rigid transfor-

mations meaning relative distances, angles, scalings and translations are preserved.

A.4 GRÖBNER

A common approach to solving systems of multivariate polynomial equations is by use of Gröbner basis functions. The solutions of a multivariate polynomial system, S, is a set of points $Q = \{Q_1, Q_2, ...Q_m\}$. One can imagine there is another polynomial system, T, that consists of different polynomials but describes the same solution set Q. The Gröbner basis is such a set with the additional advantage that in general it is easier to solve than the original system.

Say the original system S contains polynomials in variables $V = \{v_1, v_2, .., v_n\}$. The system is then formed by polynomials in the variables

$$S = \begin{cases} v_1, v_2, ..., v_n \\ v_1, v_2, ..., v_n \\ \quad ... \\ v_1, v_2, ..., v_n \end{cases} . \tag{A.4.1}$$

For solving a polynomial system the lexicographic ordered Gröbner basis is typically used. This Gröbner basis, T, has equal number of polynomials and variables. The basis is ordered such that the number of variables increases

$$T = \begin{cases} \quad v_1 \\ v_1, v_2 \\ \quad .. \\ v_1, v_2, ..., v_n \end{cases} . \tag{A.4.2}$$

Solving system T is relatively straightforward. First we solve v_1 using the first polynomial in T. This can be done using any univariate root solver, for example Laguerre's method. The second polynomial is then used to solve v_2 by first substituting the solution of v_1 into the second polynomial. The process is repeated until all variables are solved. This "forward substitution" is very similar to Gaussian elimination of a linear system.

The process is repeated until all roots are computed. The set of solutions solves both the Gröbner system T as the original system S.

Complexity

Although every polynomial system has a Gröbner basis, computing the basis may be infeasible for equations with many free and quantifier variables.

The computation complexity of finding the Gröbner basis for a system of polynomials increases significantly with both the number of variables and the degree of the polynomial. This is because both factors increase the number of solutions of the problem, and the worst case time complexity

is double exponential with respect to the number of solutions. In practice though computation may be a lot quicker.

The degree of a polynomial is the highest degree off all monomials. The degree of an individual monomial is the sum of the exponents over the variables. For example, the degree of a multivariate polynomial $x^3y^2 + 2y^3$ is 5 (the combined power of x^3 and y^2). For a univariate polynomial the degree is simply the highest power of the free variable.

Example

Let us look at an example. We intend to find the roots of a relatively simple bivariate system of polynomial equations. The system is shown in figure A.4.1a and is defined by the equations

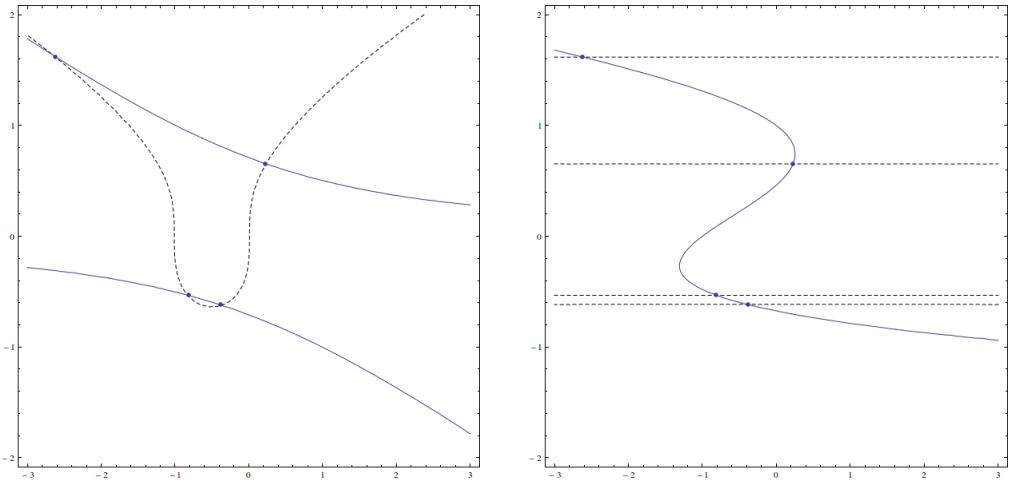
$$\begin{aligned} p1 &= x^2 + x - y^3 = 0 \\ p2 &= xy + 2y^2 - 1 = 0 \end{aligned} . \tag{A.4.3}$$

For the example only 2 variables are used and equations 1 and 2 are of degree 3 and 2 respectively. Because of the low degree and number of variables the example system can easily be reduced to its Gröbner basis. The lexicographic basis for this example is

$$\begin{matrix} -1 - y + 4y^2 + 2y^3 - 4y^4 + y^5 & \{y\} \\ 1 + x - 2y - 2y^2 + 4y^3 - y^4 & \{x, y\} \end{matrix} , \tag{A.4.4}$$

where the { } brackets indicate the variables of the equation. The polynomials of this Gröbner basis differ much from the original equations (see differences between figure A.4.1a and A.4.1b). Note however that that the roots of both systems are the same. This is visible in the figure as both systems share the same intersection points.

As seen, the benefit of a Gröbner basis is that the system can be solved by forward substitution. For the presented example this results in 4 solutions for the variable pair$x$ and y. The solutions are marked in figure A.4.1 with dots.



(a) A set of polynomials $p1$ and $p2$. $p1$ is dashed and $p2$ is solid.

(b) A Gröbner basis having identical roots to the polynomials from figure A.4.1a. The dashed and solid lines represent the first and second basis equation respectively.

Figure A.4.1: Roots for example system of polynomial equations.

Basis computation

Computation of the Gröbner basis is usually done using Buchberger original algorithm [5]. The algorithm uses multivariate division for a predefined monomial ordering to compute the basis of a set polynomial equations. A *monomial* is a single term of a polynomial, i.e. the product of one or more of the variables in the equation. Examples of monomials are: 1, x^2,xy and x^2yz.

Many different monomial *orderings* exist. The efficiency of the multivariate division is greatly influenced by the ordering imposed. Let $M(f) = M(f_1, \ldots, f_n)$ be the monomials for the system of polynomial equations f. The total ordering of the monomials is chosen to have the following two properties

1. $1 \leq t \; for \; t \in M$.
2. If $t_1 \leq t_2$ then $t_1 m \leq t_2 m$ where $t_1, t_2, m \in M$. i.e. The ordering respects multiplication.

The most frequently used ordering for elimination is Lexicographical ordering. Lexicographical ordering was originally presented by Buchberger. Terms are order lexicographically (alphabetically) and individual exponents are ordered decreasing. For example a Lexicographical ordering of the set $\{x^3, y^3, x^2, xy^2, y^4, xyz\}$ is $\{x^3, x^2, xy^2, xyz, y^4, y^3\}$. The resulting system for a lexicographical ordering is easily solved using forward substitution. However computation of a Gröbner basis is generally slow as discussed earlier.

A.5 ELIMINATION THEORY

Elimination theory describes the set of method's that eliminate some variable from a multivariate set of polynomials.

In general, given s equations: $< f_1, .., f_s > \in k[x_1, ..x_n]$, $n-1$ variables can be eliminated by combining equations into a single univariate polynomial g_j. If $deg(f_i)$ is the degree of polynomial f_i then the degree of a univariate polynomial g_j in variable x_j is $deg(g_j) = \Pi_{i \in \{1..s\}} deg(f_i)$. The roots of system f_i match those of the set of equations g_j. The roots from equations g_j however are not paired. So an additional combination step is required to obtain the root pairs or sets for original system f_i from g_j roots. There are many method of doing so, in this work we use bipartite graph matching described by Bekker et. al. [3].

Let look at an example system of two bivariate polynomials

$$\begin{cases} f_1(x_1, x_2) & = x_1^2 - 2x_2x_1 + 3x_2 - 10 = 0 \\ f_2(x_1, x_2) & = x_2^2 + x_1x_2 - x_1 - 5 = 0 \end{cases}. \tag{A.5.1}$$

The polynomials f_1 and f_2 are both of degree 2 and thus a total of 4 complex or real solutions exist. Using elimination this system is reduced to two univariate polynomials of degree $deg(f_1) * deg(f_2) = 4$

$$g_1(x_1) = 3x_1^4 - 7x_1^3 - 48x_1^2 + 81x_1 - 55 = 0\,, \tag{A.5.2}$$
$$g_2(x_2) = 3x_2^4 + x_2^3 - 36x_2^2 + 33x_2 - 15 = 0\,. \tag{A.5.3}$$

Both univariate polynomials of degree 4 have 4 roots which are equal to the roots of f_1 and f_2. However the roots found from the equations g_1 and g_2 are disconnected. Table 7 shows the numerical roots of both the original system and the set of univariate polynomials. Notice that the roots from system f and disjoint equations g match. For this example using the Mathematica numerical solver the univariate roots even seem to be paired. However it is just as likely that the univariate roots are found in a different order. As mentioned, in order to combine the univariate roots an additional combination step would is still required.

x	y
-3.66667	-0.333333
2.16745	-3.97196
4.36147	1.57653
-0.528918	2.39543

(a) System f_1 and f_2 roots.

x	y
-3.66667	-0.333333
-0.528918	-3.97196
2.16745	1.57653
4.36147	2.39543

(b) Roots g_1.

(c) Roots g_2.

Table 7: Roots of example system.

A.6 LAGUERRE ROOTS SOLVING

Laguerre's method is a numerical root-finding algorithm for univariate polynomials. The method is known for nearly always finding a root given an initial guess. Although Laguerre's method only finds a single root, it is often used to solve all roots of a univariate polynomial. Intermediate roots are used to simplify/deflate the polynomial by one degree. Both real and complex roots are found one by one and are used to simplify the polynomial until algebraic root expressions can be applied. Depending on the implementation roots for fourth or third degree polynomials are algebraically calculated.

In our discussion we will use Laguerre root (singular) solving method to indicate the application of the method to find a single root. The Laguerre roots (plural) finding method repeatedly uses the root solving method to obtain all roots of a univariate polynomial.

Root finding

The single root finding method is presented in algorithm A.1. It is a numerical method requiring a initial guess, a maximum error C_1 and an upper iteration limit C_2. Given a polynomial p of degree n, each iteration G, H and a are calculated as follows

$$G = \frac{p'(x_k)}{p(x)}, \tag{A.6.1}$$

$$H = G^2 - \frac{p''(x_k)}{p(x_k)}, \tag{A.6.2}$$

$$a = \frac{n}{G \pm \sqrt{(n-1)(nH - G^2)}}. \tag{A.6.3}$$

Laguerre's method uses both the first and second derivative yielding a fast convergence. The method convergences cubic to single roots. In case of multiple roots (duplicate roots) the method converges slower, only linearly.

Algorithm A.1 Laguerre single root finding method.

x_0=initial guess, $a = \infty$, $k = 0$
while $a > C_1$ and $k \leq C_2$
  calculate G
  calculate H with G
  calculate a with H and G
  $x_{k+1} = x_k - a$
  $k = k + 1$
endwhile
root = x_k

Roots finding

Laguerre's roots finding method is again an iterative approach where each step the polynomial $P = C\,(x - x_1)(x - x_2)...(x - x_N)$ is altered. Given a root x_i the simplified or deflated polynomial P' is defined as

$$P = (x - x_i) * P'(x), \tag{A.6.4}$$

where $deg(P') = deg(P) - 1$.

The deflation process introduces a numerical error because of numerical precision. These errors are minimal if the roots are found in increasing order, i.e. $x_i \leq x_j$ where i is found before j. In addition a root polishing step can be used to improve numerical precision. Here a root x is found by the Laguerre root solver. x is then used as an initial guess for the Laguerre root solver to find root x' which is more precise than x. The entire process is seen in algorithm A.2.

Algorithm A.2 Laguerre roots finding method.

$roots = \{\}$
while $deg(p) > 3$
  guess = 0
  $x = Laguerrerootsolve(p, guess)$
  $x' = Laguerrerootsolve(p, x)$
  $roots = roots \cup x'$
  $p = deflate(p, x')$
endwhile
$roots$= $roots \cup$ find_roots_algebraic(p)

A.7 POINT SETS

POINT SET 1 [8]

```
30 0.001475 7.892267
1.05636 -29.981396 27.519008
-29.366428 -6.132937 13.137551
28.698913 8.739131 40.731883
-12.893256 -27.088078 56.081574
-22.315616 20.050269 31.164982
14.611799 -26.201056 2.074327
28.323328 9.888835 31.782012
-14.26238 -26.39288 0.461891
-22.304724 20.062385 4.010534
-26.057635 14.866056 41.206363
-25.432673 -15.911603 55.826190
17.043966 -24.688119 31.615727
25.129457 16.386287 39.235138
-25.917641 15.108801 42.071436
25.362190 -16.03271 45.731882
-2.34494 29.908214 2.847871
-2.62016 -29.88536 19.694054
-20.170149 -22.206503 45.384629
29.952444 -1.68852 21.92032
30.001 0.001475 7.892267
1.056395 -29.982395 27.519008
-29.367407 -6.133141 13.137551
28.699869 8.739422 40.731883
-12.893685 -27.088981 56.081574
-22.316359 20.050938 31.164982
14.612286 -26.201929 2.074327
28.324273 9.889165 31.782012
-14.262855 -26.393767 0.461891
-22.305468 20.063053 4.010534
-26.058504 14.866552 41.206363
-25.433521 -15.912134 55.82619
17.044534 -24.688942 31.615727
25.130294 16.386834 39.235138
-25.918505 15.109304 42.071436
25.363036 -16.024245 45.731882
-2.345018 29.909211 2.847871
-2.620248 -29.886356 19.694054
-20.171721 -22.207243 45.384629
29.953442 -1.688577 21.92032
```

POINT SET 2 [8]

```
60.051121 0.002953 3.946134
-57.932024 15.399312 15.983017
57.432130 17.488707 20.365942
55.022756 -23.936632 11.505062
29.180100 -52.423113 1.037163
-58.861558 -11.113569 20.134482
-44.597179 40.113733 2.005267
-23.247383 -55.406652 17.669299
34.041568 -49.309081 15.807863
-34.084135 -49.427745 12.479981
50.684216 -32.022045 22.865941
57.318676 17.619539 22.082457
-40.408130 -44.485701 22.692315
-39.838370 44.994386 7.411167
-10.261352 -59.146784 22.600675
53.919844 26.493193 18.949042
-8.540012 59.442972 13.092342
-59.369089 8.361285 7.133233
-38.029817 46.404843 4.995216
47.946099 -35.925380 27.276243
```

POINT SET 3 [8]

```
-11.820859 50.421254 -15.817382
42.403448 -6.693162 56.567707
10.366902 80.249947 26.965969
18.527457 61.577469 -13.680418
23.930322 23.878386 -41.820643
66.363729 0.636729 49.246025
-3.608026 -24.493246 39.678687
75.507564 20.208045 6.298139
48.919097 55.614254 -13.266609
65.713317 2.841028 3.498858
46.632786 80.517454 4.866333
13.598993 83.519129 30.375
84.570573 18.219363 28.224203
2.322453 -10.802862 51.268799
82.820384 38.516367 9.148307
3.553158 75.111087 30.738097
-5.898713 21.39033 60.097056
30.009532 -24.696147 35.870356
-3.793621 -14.263808 46.897322
```

```
58.357492 87.161327 11.960644
33.207329 64.844079 -10.665479
34.46129 41.806234 94.623903
-26.871029 3.103967 39.48246
-4.153639 67.427229 23.451422
22.371000 47.845956 88.060867
67.398986 16.520701 79.062822
79.257377 49.418921 4.727043
-37.543275 31.718373 8.573268
49.576671 65.965076 -6.501629
96.781947 53.421231 22.908004
-18.623157 23.988046 47.691608
58.416292 -4.557784 48.525368
48.408528 15.833662 81.511728
31.694971 -2.169579 63.538387
-18.366214 2.837799 46.415679
81.087477 11.573666 46.319607
57.311572 -9.09605 38.123767
68.59397 33.580936 -6.118165
89.036231 21.72231 35.086999
3.141412 52.730721 67.919265
```

POINT SET 4 [8]

```
11.094300 0.452200 65.232800
5.094000 10.845000 65.076500
-6.906300 10.843900 65.008900
-12.906500 0.449800 65.089700
-6.906300 -9.942900 65.054000
5.094000 -9.941800 65.221600
10.954600 0.522000 75.231600
4.954400 10.914800 75.075200
-7.045900 10.913700 75.077000
-13.046100 0.519600 74.896400
-7.045900 -9.873100 75.052800
4.954470 -9.872000 75.220400
10.815000 0.591800 85.230400
4.814800 10.984600 85.074000
-7.185500 10.983500 85.064100
-13.185800 0.589400 84.895200
-7.185500 -9.803300 85.051600
4.814900 -9.802200 85.217100
10.675400 0.661600 95.229100
4.675200 11.054400 95.072800
```

```
-7.325300 11.053300 94.907700
-13.325400 0.659200 95.094000
-7.325200 -9.733500 95.050400
4.675200 -9.732300 95.217900
```

BIBLIOGRAPHY

[1] Pankaj K Agarwal and Micha Sharir. Efficient randomized algorithms for some geometric optimization problems. *Discrete & Computational Geometry*, 16(4):317–337, 1996.

[2] G.T. Anthony, H.M. Anthony, B. Bittner, B.P. Butler, M.G. Cox, R. Drieschner, R. Elligsen, A.B. Forbes, H. Gross, S.A. Hannaby, P.M. Harris, and J. Kok. Reference software for finding Chebyshev best-fit geometric elements. *Precision Engineering*, 19(1):28–36, 1996. ISSN 0141-6359. doi: 10.1016/0141-6359(96)00005-0. URL http://www.sciencedirect.com/science/article/pii/0141635996000050.

[3] H Bekker, EP Braad, and Boris Goldengorin. Using bipartite and multidimensional matching to select the roots of a system of polynomial equations. In *Computational Science and Its Applications–ICCSA 2005*, pages 397–406. Springer, 2005.

[4] O. Bottema and G.R. Veldkamp. On the lines in space with equal distances to n given points. *Geometriae Dedicata*, 6(1):121–129. ISSN 0046-5755. doi: 10.1007/BF00181587. URL http://dx.doi.org/10.1007/BF00181587.

[5] B. Buchberger. A Theoretical Basis for the Reduction of Polynomials to Canonical Forms. *SIGSAM Bull.*, 10(3):19–29, August 1976. ISSN 0163-5824. doi: 10.1145/1088216.1088219. URL http://doi.acm.org/10.1145/1088216.1088219.

[6] Kirsten Carr and Placid Ferreira. Verification of form tolerances part II: Cylindricity and straightness of a median line. *Precision Engineering*, 17(2):144–156, 1995. ISSN 0141-6359. doi: 10.1016/0141-6359(94)00018-U. URL http://www.sciencedirect.com/science/article/pii/014163599400018U.

[7] Thomas Chaperon and François Goulette. Extracting Cylinders in Full 3D Data Using a Random Sampling Method and the Gaussian Image. In Thomas Ertl, Bernd Girod, Heinrich Niemann, and Hans-Peter Seidel, editors, *VMV*, pages 35–42. Aka GmbH, 2001. ISBN 3-89838-028-9. URL http://dblp.uni-trier.de/db/conf/vmv/vmv2001.html#ChaperonG01.

[8] S Hossein Cheraghi, Guohua Jiang, and Jamal Sheikh Ahmad. Evaluating the geometric characteristics of cylindrical features. *Precision Engineering*, 27(2):195–204, 2003.

[9] A. Clement and P. Bourdet. A Study of Optimal-Criteria Identification Based on the Small-Displacement Screw Model. *{CIRP} Annals - Manufacturing Technology*, 37(1):503–506, 1988. ISSN 0007-8506. doi: 10.1016/S0007-8506(07)61687-4. URL http://www.sciencedirect.com/science/article/pii/S0007850607616874.

[10] Olivier Devillers and Franco P Preparata. Evaluating the cylindricity of a nominally cylindrical point set. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 518–527. Society for Industrial and Applied Mathematics, 2000.

[11] Lowell W Foster. *Geo-metrics III: The Application of geometric tolerancing techniques (using the customary inch system): As based upon harmonization of national and international standards practices*. Addison-Wesley Publishing Company, 1994.

[12] G Goch and K Lübke. Tschebyscheff approximation for the calculation of maximum inscribed/minimum circumscribed geometry elements and form deviations. *CIRP Annals-Manufacturing Technology*, 57(1):517–520, 2008.

[13] THOM J Hodgson, MICHAEL G Kay, RAVI O Mittal, and SHIH-YI Tang. Evaluation of cylindricity using combinatorics. *IIE transactions*, 31(1):39–47, 1999.

[14] Michael E. Houle and Godfried T. Toussaint. Computing the Width of a Set. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(5): 761–765, 1988. URL http://dblp.uni-trier.de/db/journals/pami/pami10.html#HouleT88;http://doi.ieeecomputersociety.org/10.1109/34.6790;http://www.bibsonomy.org/bibtex/2011669037aefb7863a7d819f954e6816/dblp.

[15] Jyunping Huang. An exact minimum zone solution for three-dimensional straightness evaluation problems. *Precision Engineering*, 23(3):204–208, 1999.

[16] Jyunping Huang. An efficient approach for solving the straightness and the flatness problems at large number of data points. *Computer-Aided Design*, 35(1):15–25, 2003. URL http://dblp.uni-trier.de/db/journals/cad/cad35.html#Huang03;http://dx.doi.org/10.1016/S0010-4485(01)00172-5;http://www.bibsonomy.org/bibtex/2c54fc6005b834bad4150b68f6ddb674e/dblp.

[17] ST Huang, KC Fan, and John H Wu. A new minimum zone method for evaluating straightness errors. *Precision engineering*, 15(3):158–165, 1993.

[18] Hsin-Yi Lai, Wen-Yuh Jywe, Cha'o-Kuang Chen, and Chien-Hong Liu. Precision modeling of form errors for cylindricity evaluation using genetic algorithms. *Precision Engineering*, 24(4):310–319, 2000.

[19] Moon-Kyu Lee. An enhanced convex-hull edge method for flatness tolerance evaluation. *Computer-Aided Design*, 41(12):930–941, 2009. URL http://dblp.uni-trier.de/db/journals/cad/cad41.html#Lee09a; http://dx.doi.org/10.1016/j.cad.2009.06.011;http://www.bibsonomy.org/bibtex/242a2a6b41621f2969445c5cdefce1cfb/dblp.

[20] Daniel Lichtblau. Cylinders through five points: Computational Algebra and Geometry. *Journal of Mathematics Research*, 4(6):p65, 2012.

[21] U. Roy and X. Zhang. Establishment of a pair of concentric circles with the minimum radial separation for assessing roundness error. *Computer-Aided Design*, 24(3):161–168, 1992. ISSN 0010-4485. doi: 10.1016/0010-4485(92)90035-9. URL http://www.sciencedirect.com/science/article/pii/0010448592900359.

[22] Utpal Roy and Yaoxian Xu. Form and orientation tolerance analysis for cylindrical surfaces in computer-aided inspection. *Computers in Industry*, 26(2):127–134, 1995. ISSN 0166-3615. doi: 10.1016/0166-3615(94)00031-K. URL http://www.sciencedirect.com/science/article/pii/016636159400031K.

[23] Xiangchao Zhang, Xiangqian Jiang, and Paul J Scott. A reliable method of minimum zone evaluation of cylindricity and conicity from coordinate measurement data. *Precision Engineering*, 35(3):484–489, 2011.