Saurav Agarwal
Robotics Laboratory,
Department of Engineering Design,
Indian Institute of Technology Madras,
Chennai 600 036, India
e-mail: agr.saurav1@gmail.com

Rangaprasad Arun Srivatsan[1]
Robotics Institute,
Carnegie Mellon University,
5000 Forbes Avenue,
Pittsburgh, PA 15213
e-mail: rarunsrivatsan@cmu.edu

Sandipan Bandyopadhyay[2]
Robotics Laboratory,
Department of Engineering Design,
Indian Institute of Technology Madras,
Chennai 600 036, India
e-mail: sandipan@iitm.ac.in

Analytical Determination of the Proximity of Two Right-Circular Cylinders in Space

This paper presents a novel analytical formulation for identifying the closest pair of points lying on two arbitrary cylinders in space, and subsequently the distance between them. Each cylinder is decomposed into four geometric primitives. It is shown that the original problem reduces to the computation of the shortest distance between five distinct combinations of these primitives. Four of these subproblems are solved in closed form, while the remaining one requires the solution of an eight-degree polynomial equation. The analytical nature of the formulation and solution allows the identification of all the special cases, e.g., positive-dimensional solutions, and the curve of intersection when the cylinders interfere. The symbolic precomputation of the results leads to a fast numerical implementation, capable of solving the problem in about 50 μs (averaged over 1×10^6 random instances of the most general case) on a standard PC. The numerical results are verified by repeating all the calculations in a general-purpose commercial CAD software. The algorithm has significant potential for applications in the various aspects of robotics and mechanisms, as their links can be modeled easily and compactly as cylinders. This makes tasks such as path planning, determination of the collision-free workspace, etc., computationally easier. [DOI: 10.1115/1.4032211]

1 Introduction

The cylindrical shape has a significant importance in multiple fields of engineering and technology, such as robotics, mechanisms, computer-aided design and manufacturing, and computer graphics. Approximately cylindrical shape is prevalent among the common geometries of the links of robots, e.g., the serial robots KUKA KR5-2 arc HW[3] used for welding, FANUC M-900**i**A[3] used for assembly in the industries, or parallel robots, e.g., FANUC M-1**i**A[3], FANUC 200**i**B[4]. The same is true in the case of mechanisms as well, as shown in Ref. [1]. The cylindrical shape is important in other fields of engineering as well, e.g., in designing a pipeline layout [2], in determining tool collisions during a machining operation [3], or in the design of temperature control systems for molds [4].

In early attempts to find the shortest distance between a pair of approximately cylindrical links of a robot, the links were abstracted as line segments, and the distances between these segments were computed [5,6]. This is equivalent of shrinking the cylindrical-shaped link radially till it coincides with the line segment forming the axis of the cylinder. The geometric model is closer to the reality in Ref. [7], in which the cylinder retains its original dimensions, but the end caps are no longer disks—they are replaced by hemispheres. This modification is significant from a computational perspective. A cylinder can be thought of as having an infinite extent in the axial dimension, limited only by two disk-shaped "end caps." However, the disks themselves are parts of two planes, and as such, they would have infinite extents too, except for the bounding circles, at which they intersect the curved surface of the cylinder (see Fig. 1(*a*)). The hemisphere, on the other hand, is a subset of a closed and finite surface, and is therefore easier to handle mathematically. However, the "critical distance" computed on the basis of this idealization of a cylinder does not necessarily coincide with the shortest distance between the actual cylinders.

More recent works retain the actual geometry of the cylinders. Consequently, they have to deal with *all* the *relevant* scenarios that can occur. This invariably leads to algorithmic complexity, as well as demanding computations. For instance, in Ref. [4], each cylinder is decomposed into four simple geometric primitives, viz., a line segment representing the axis, the cylindrical surface, and a pair of disk and circle, forming each of the end caps. The computation of *proximity* of (i.e., the least distance between) a pair of cylinders is broken down to the computation of the proximity of various pairs: once between two cylindrical surfaces, eight times between circles and disks, four times between circles and line segments, and four times between pairs of circles. These computations are *logically redundant*, and hence, the efficiency of the algorithm is compromised.

A different approach, based on the line geometry and dual number algebra, is adapted in Ref. [1]. In this work, the actual geometry of the cylinders is considered and the final result is obtained by an algorithm covering seven possible scenarios in a hierarchical manner. A more recent work based on a similar approach is reported in Ref. [8].

The study of the existing works reveals several avenues for improvement over the current state-of-the-art. First, the computations should ideally be analytical in nature, to the extent possible, not only to avoid potentially erroneous results stemming out of numerical inaccuracies/approximations but also to harvest the benefit of symbolic precomputations to speed up the subsequent numerical implementations. Next, all the reports cited above focus on resolving the binary issue, i.e., whether two cylinders are in interference or not, and except for Ref. [4], none of them compute the proximity of the cylinders when they are *not* in interference. Finally, none of the above compute the pair of *proximal points* (i.e., the pair of points delimiting the shortest distance). In a real-life scenario, e.g., in the case of the offline validation of a path in a robot simulation software, it is of immense value to the user to know which part(s) of the robot come(s) closest to the other part(s) of the robot, or objects in its environment. This knowledge, coupled with the direction in which the distance between the two objects is the closest (given by the proximal points), helps the user to decide about the required modifications in the path to steer clear

[1]The author contributed to this work as a Project Officer working in the Robotics Laboratory, Department of Engineering Design, Indian Institute of Technology Madras.
[2]Corresponding author.
[3]KUKA Robotics: http://www.kuka-robotics.com
[4]FANUC Robotics: http://www.fanucrobotics.com

Manuscript received August 11, 2015; final manuscript received November 23, 2015; published online March 8, 2016. Assoc. Editor: David Dooner.

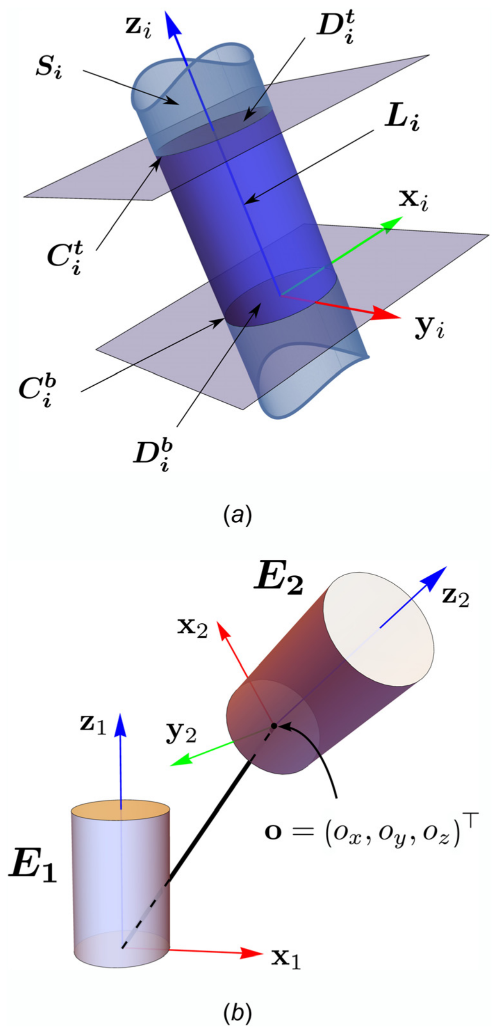

Fig. 1 (*a*) The cylinder E_i and its constituent geometric primitives—the cylindrical surface S_i; bottom end cap D_i^b, top end cap D_i^t, and their bounding circles, C_i^b, C_i^t, respectively; the axis, L_i. (*b*) The relative position and orientation of the two cylinders. The local coordinate system of the first cylinder serves as the global frame of reference.

of impending collisions downstream in a path, or to simply increase the safety margin otherwise. In a different scenario, such information can be used in an offline/real-time manner to achieve algorithmic path planning for a robot, with obstacle avoidance.

This paper attempts to address the above-mentioned issues. It presents an analytical formulation to solve the problem of finding the proximity of two arbitrary cylinders in space in a complete manner. Not only it finds the proximity and the pair of *proximal points*, which define this distance, but it also reports interference as and when it occurs. In addition, it addresses *all* the special cases, such as computing the two lines defining the set of proximal points when the axes of the cylinders are parallel; two patches on the disks at the ends when they directly face each other or coincide, in the case of two cylinders placed end-to-end, and so on. This is achieved by a decomposition of each cylinder into four geometric primitives, and then finding the shortest distance between a few pairs among them in the symbolic form. As shown by the authors in Ref. [9], distance queries between only five distinct pairs of primitives, when arranged in a logical sequence, suffice to solve the original problem in its completeness. To compute the distance between each of the pairs of primitives, the primitives are represented parametrically, and the proximal points are found in the parameter space by solving nonlinear equations arising out of the minimization of the distance function with respect to the said parameters. The orientation of one cylinder relative to the other is represented in terms of rotation matrices, which in turn are parametrized in terms of *unit quaternions*. This allows natural inclusion of all the special cases in the formulation, such as the case of two parallel cylinders. To the best of the authors' knowledge, this approach is completely novel in regard to the problem. Owing to the symbolic nature of the formulation, the solutions are obtained in *closed form* in all the cases, except in the case of a pair of circles, in which case the problem reduces to the solution of a polynomial equation of degree eight, as reported earlier in Ref. [10]. Nevertheless, the coefficients of the said polynomial are obtained in closed form, which allows for fast numerical computation of its roots.

Utilizing the analytical results precomputed in the commercially available computer algebra system, MATHEMATICA,[5] the numerical implementation in C++ computes the proximity and the proximal points for about 20,000 random pairs of cylinders every CPU-second, making it ideal for inclusion in a real-time[6] application. It can also be used to scan the design space or configuration space of robots for self-collisions, which is an essential step in the computation of the *safe working zone* (as defined in Ref. [11]) of a robotic manipulator, or in planning the paths of hyper-redundant robots, such as the "snake robots" [12,13].

The rest of the paper is organized as follows: The formulation for computing the proximity and identifying the proximal points is explained in detail in Sec. 2. The details of the decomposition of the cylinder into four primitives and the corresponding decomposition of the original problem into different proximity queries among these primitives are presented in Sec. 3. The formulation of the proximity query for each of the above-mentioned pairs, and their solutions, including the special cases is documented. Section 4 presents the numerical examples. In particular, an example of the computationally worst-case scenario is described in detail. The validation of the final results produced by the algorithm and an analysis of the same are presented in Sec. 5. Finally, the paper is concluded in Sec. 6.

2 Formulation of the Overall Problem

In this section, the algorithm for finding the proximity and the proximal points is discussed. Branching of the overall problem into subcases is presented via algorithmic flowcharts and illustrated via figures depicting the geometric scenario in each subcase. The first step in the formulation is to describe a solid cylinder in terms of four distinct types of geometric primitives, namely, line segment, circle, disk, and cylindrical surface. For the *i*th cylinder, these are shown in Fig. 1(*a*). The curved surface, S_i, has $\mathbf{z}_i$ as its axis and is extended to infinity along this direction. This surface is clipped by two planes perpendicular to $\mathbf{z}_i$, generating a *finite* cylinder E_i of length l_i between them. The line segment forming the axis of the finite cylinder is denoted by L_i. At the top end, the clipping plane intersects S_i to form a circle C_i^t of radius r_i, which bounds the disk-shaped end cap D_i^t. Similarly, at the bottom end, D_i^b is bound by C_i^b. A local frame of reference is attached to the center of D_i^b (or C_i^b).

The proximity of a pair of these primitives, say, A and B, is defined by

$$P(A,B) = \min(d(A,B)) \qquad (1)$$

[5]Wolfram Mathematica, version 10.2: http://www.wolfram.com/mathematica/

[6]The authors are aware of numerical libraries/tools dedicated to the computation of the distances between two surfaces/solids, e.g., Bullet Physics Library (http://bulletphysics.org/), Proximity Query Package (http://gamma.cs.unc.edu/SSV/), and they have verified, e.g., that Bullet (version 3.0) routines can be significantly faster than the proposed algorithm. However, they wish to point out that the main contribution of the present work lies in its analytical nature, and while the proposed algorithm is not the fastest available, it is nevertheless suitable in a demanding real-time scenario.

where

$$d(A,B) = \|\mathbf{p}_1 - \mathbf{p}_2\|, \quad \mathbf{p}_1 \in A, \quad \mathbf{p}_2 \in B \tag{2}$$

Designating the pair of proximal points as $\mathbf{p}_1^* \in A$ and $\mathbf{p}_2^* \in B$, one can write

$$P(A,B) = \|\mathbf{p}_1^* - \mathbf{p}_2^*\| \tag{3}$$

The above equations hold good, irrespective of whether the two cylinders are *external* to each other or one appears *inside* the other, e.g., as shown in Fig. 2. Therefore, to complete the physical interpretation of the final solution, one has to check for the *inclusion* of one cylinder into the other. This can be done easily by verifying if $\mathbf{p}_1^*$ lies inside E_2 or $\mathbf{p}_2^*$ inside E_1. However, for the rest of the paper, it is assumed that the cylinders are external to each other.

The shortest distance and the proximal points can be obtained with the help of five independent distance queries (detailed in Sec. 3), namely, the shortest distance between two cylindrical surfaces designated by—$P(S_i, S_j)$, a disk and a line—$P(D_i, L_j)$, a circle and a line—$P(C_i, L_j)$, a disk and a circle—$P(D_i, C_j)$, and two circles—$P(C_i, C_j)$. The flowchart in Fig. 3 portrays the branching of original problem into three different subproblems, based on the proximal points obtained from $P(S_1, S_2)$, which form the overall hierarchy of the entire algorithm. The algorithm for finding $P(D_i, S_j)$ is depicted in the form of a flowchart in Fig. 4. Similarly, the flowchart depicting the algorithm for computation of $P(D_i, D_j)$ is presented in Fig. 5. The algorithm has a possibility of termination at every decision level. Therefore, it is computationally efficient, since not all cases need to flow down to the last level (see Sec. 3 for more details on this). The flowcharts show that the algorithm posses only three different levels of checks in the worst-case scenario. It is evident that the hierarchy is compact, and consequently, the algorithm is simpler to implement than those reported in Refs. [1,7]. The following are the steps to obtain $P(E_1, E_2)$ in terms of the geometric primitives:

— As shown in Fig. 3, the algorithm starts with finding $P(S_1, S_2)$

$$P(E_1, E_2) = P(S_1, S_2) \tag{4}$$

The formulation to compute the same is described in Sec. 3.

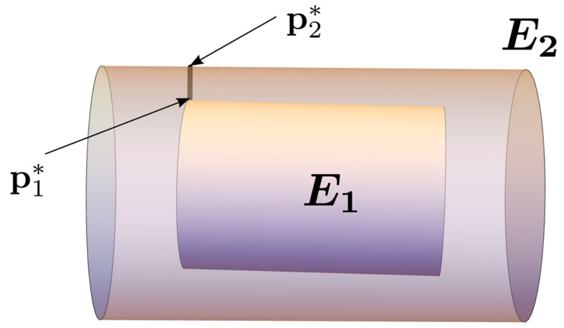

Fig. 2 One cylinder inside the other cylinder

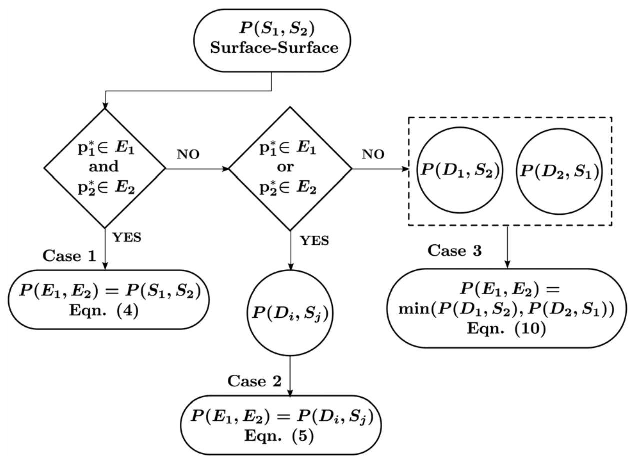
$P(S_1,S_2)$ Surface-Surface
$\mathbf{p}_1^* \in E_1$ and $\mathbf{p}_2^* \in E_2$
NO
$\mathbf{p}_1^* \in E_1$ or $\mathbf{p}_2^* \in E_2$
NO
$P(D_1,S_2)$
$P(D_2,S_1)$
YES
Case 1
$P(E_1,E_2) = P(S_1,S_2)$ Eqn. (4)
YES
$P(D_i,S_j)$
Case 2
$P(E_1,E_2) = P(D_i,S_j)$ Eqn. (5)
Case 3
$P(E_1,E_2) = \min(P(D_1,S_2), P(D_2,S_1))$ Eqn. (10)

Fig. 3 Flowchart of the algorithm at the first level depicting the branching of the original problem into three subproblems. The final results are obtained via the equations given in each terminal block.

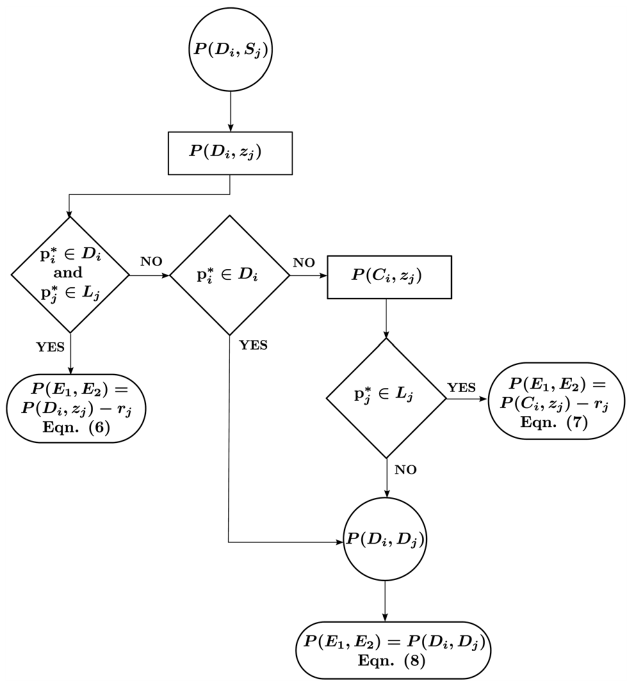
$P(D_i,S_j)$
$P(D_i,z_j)$
$\mathbf{p}_i^* \in D_i$ and $\mathbf{p}_j^* \in L_j$
NO
$\mathbf{p}_i^* \in D_i$
NO
$P(C_i,z_j)$
YES
$P(E_1,E_2) = P(D_i,z_j) - r_j$ Eqn. (6)
YES
$\mathbf{p}_j^* \in L_j$
YES
$P(E_1,E_2) = P(C_i,z_j) - r_j$ Eqn. (7)
NO
$P(D_i,D_j)$
$P(E_1,E_2) = P(D_i,D_j)$ Eqn. (8)

Fig. 4 Flowchart of the algorithm for finding proximity of a disk and a cylindrical surface. The final results are given by the equations mentioned in each terminal block.

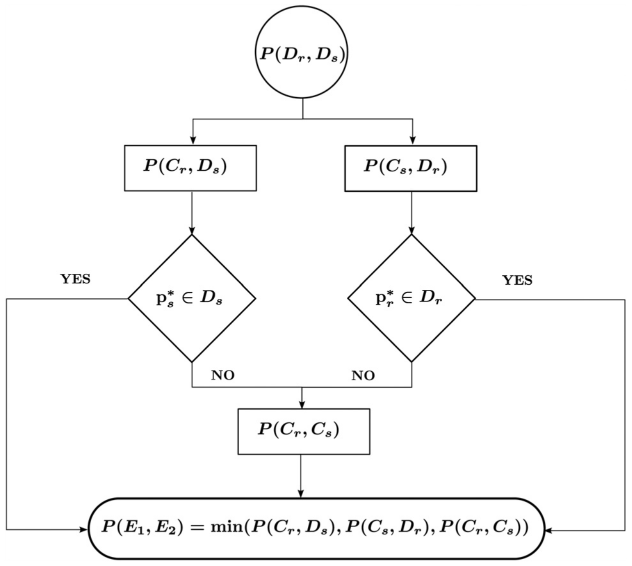

Fig. 5 Flowchart of the algorithm for finding proximity of two disks

— Since S_1, S_2 are theoretically *infinite*, the proximal points, $\mathbf{p}_1^* \in S_1$, $\mathbf{p}_2^* \in S_2$, could fall beyond the actual extent of the cylinders, i.e., it may so happen that $\mathbf{p}_i^* \in S_i$, but $\mathbf{p}_i^* \notin E_i$, for one or both of the cylinders. This observation gives rise to three possible cases:

- Case 1: If both $\mathbf{p}_1^*, \mathbf{p}_2^*$ are on the cylinders, i.e., $\mathbf{p}_1^* \in E_1$ and $\mathbf{p}_2^* \in E_2$, then the situation is designated as Case 1. No further queries are required and the algorithm terminates successfully. Such a scenario is shown in Fig. 6.
- Case 2: If one of $\mathbf{p}_1^*, \mathbf{p}_2^*$ is beyond the corresponding cylinder while the other point is on it, i.e., either $\mathbf{p}_1^* \notin E_1$ or $\mathbf{p}_2^* \notin E_2$, then it is designated as Case 2. Further distance queries are made as per the algorithm, and the proximal points, $\mathbf{p}_1^*, \mathbf{p}_2^*$, are updated accordingly. Three different configurations belonging to this case are shown in Fig. 7. It should be noted that the proximal points marked in the figure correspond to the final distance query and not the initial ones given by $P(S_1, S_2)$.
- Case 3: If both the points $\mathbf{p}_1^*, \mathbf{p}_2^*$ are beyond the corresponding cylinders, i.e., $\mathbf{p}_1^* \notin E_1$ and $\mathbf{p}_2^* \notin E_2$, then such a scenario falls under Case 3. The proximal points are updated again following the algorithm (see Figs. 3–5). Figure 8 shows one such configuration.

— Case 2: One of the points among $\mathbf{p}_1^*, \mathbf{p}_2^*$, say $\mathbf{p}_2^*$, lies beyond the corresponding cylinder, E_2, and therefore, the final proximal point would not lie on the cylindrical surface of E_2, but on the end disk or end circle closest to $\mathbf{p}_2^*$, say D_2^b, as shown in Fig. 7(*a*). Therefore, the distance is given by

$$P(E_1, E_2) = P(D_2^b, S_1) \quad (5)$$

However, the perpendicular dropped from D_2^b to S_1 is orthogonal to the $\mathbf{z}_1$ axis (carrying the line segment L_1), as shown in Fig. 7(*a*), $P(D_2^b, S_1) = P(D_2^b, z_1) - r_1$.Thus, $P(D_2^b, z_1)$ is computed instead of $P(D_2^b, S_1)$, as the former is computationally less demanding. Upon computation, if $\mathbf{p}_1^* \in L_1$, then it corresponds to Case 2.1; else to Case 2.2.

— Case 2.1: The solution point, $\mathbf{p}_1^*$, corresponding to $P(D_2^b, z_1)$, lies on the line segment, L_1, i.e., $\mathbf{p}_1^* \in L_1$, as shown in Fig. 7(*a*). The shortest distance is

$$P(E_1, E_2) = P(D_2^b, z_1) - r_1 \quad (6)$$

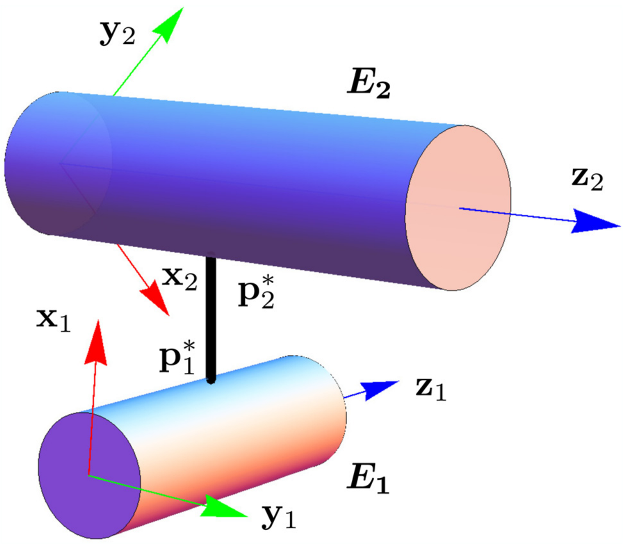

Fig. 6 Case 1.1: proximal distance and points are given by *P*(*S*, *S*)

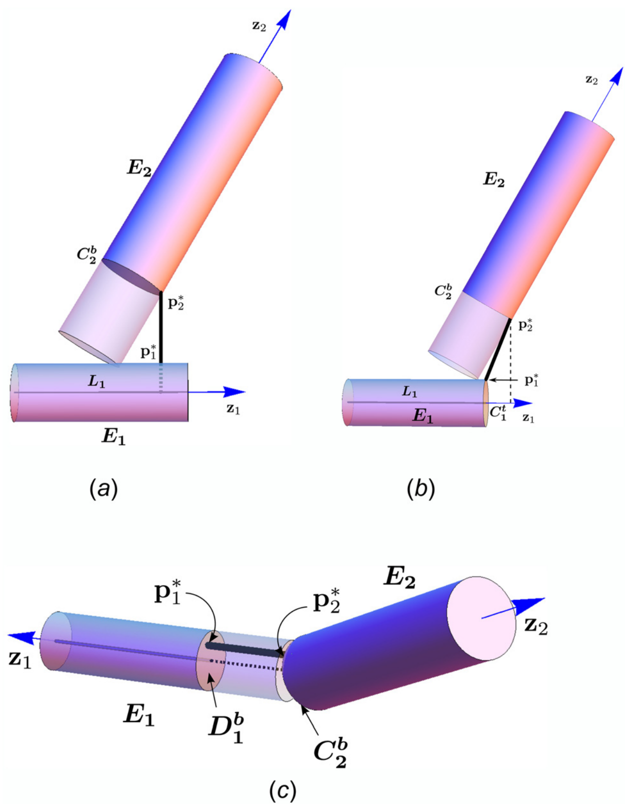

Fig. 7 Configurations illustrating Case 2. The translucent part is the theoretical extension of the actual (i.e., finite) cylinders. The initial proximal point obtained from *P*(S_1, S_2) does not lie on the first cylinder, E_1 and thus, the scenario falls under Case 2. (*a*) Case 2.1: proximity and proximal points are given by *P*(*C*, *L*). As the foot of the perpendicular drawn from C_2^b lies on L_1, it belongs to Case 2.1. (*b*) Case 2.2: proximity and proximal points are given by *P*(*C*, *C*). As the foot of the perpendicular drawn from C_2^b does not lie on L_1, it belongs to Case 2.2. (*c*) Case 2.2: proximity and proximal points are given by *P*(*D*, *C*). The foot of the perpendicular from C_2^b does not lie on L_1 and hence, it belongs to Case 2.2.

It is to be noted that in this case, the cylinders interfere if $P(D_2^b, z_1) < r_1$. If $\mathbf{p}_2^* \notin E_2$, i.e., the proximal point lies outside the boundary of the disk, D_2^b, then the shortest distance is given by

$$P(E_1, E_2) = P(C_2^b, z_1) - r_1 \quad (7)$$

The point $\mathbf{p}_1^*$ is updated to lie on the cylindrical surface using vector proportions.

— Case 2.2: The proximal point, $\mathbf{p}_1^*$, corresponding to $P(D_2^b, z_1)$, lies beyond L_1, as shown in Fig. 7(*b*). In such a situation, while the proximal point on E_2 lies on D_2^b, the proximal point on E_1 would not be on S_1, but on the disk closest to $\mathbf{p}_1^*$, say D_1^t. Following that

$$P(E_1, E_2) = P(D_1^t, D_2^b) \quad (8)$$

It is to be noted that the shortest distance between two disks always involves a point on the bounding circle of at least one of the disks. Thus, computationally less expensive $P(D, C)$ is used instead of $P(D, D)$ to obtain

$$P(D_1^t, D_2^b) = \min(P(D_1^t, C_2^b), P(D_2^b, C_1^t)) \quad (9)$$

For $P(D_1^t, C_2^b)$, if $\mathbf{p}_1^* \notin E_1$, then the proximal point $\mathbf{p}_1^*$ lies on the boundary of the disk D_1^t, i.e., the circle C_1^t. Hence, $P(C_1^t, C_2^b)$ is computed to get the proximal points. Similarly, $P(D_2^b, C_1^t)$ is evaluated. However, it may also lead to $P(C_1^t, C_2^b)$.

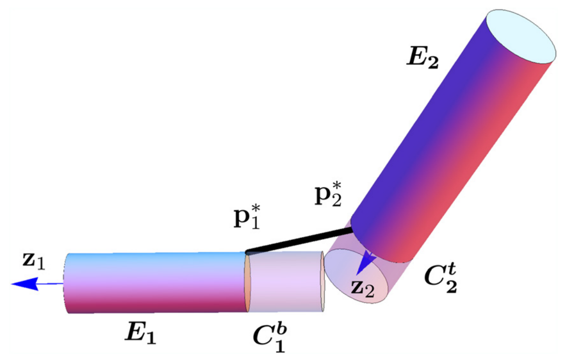

Fig. 8 Configuration corresponding to Case 3. Proximity and proximal points are given by $P(C_1^b, C_2^t)$. The translucent parts are the theoretical extensions of the actual cylinders, E_1 and E_2. The initial proximal points obtained from $P(S_1, S_2)$ lie outside their respective cylinders, resulting in the classification Case 3.

— Case 3: Both $\mathbf{p}_1^*$ and $\mathbf{p}_2^*$ corresponding to $P(S_1, S_2)$ lie beyond the actual cylinders. Say, D_1^b and D_2^b are the disks closest to the proximal points $\mathbf{p}_1^*$ and $\mathbf{p}_2^*$, respectively. In such a situation, $P(D_1^b, S_2)$ and $P(D_2^b, S_1)$ are computed and the proximity is given by

$$P(E_1, E_2) = \min(P(D_1^b, S_2), P(D_2^b, S_1)) \quad (10)$$

where $P(D_1^b, S_2)$ and $P(D_2^b, S_1)$ are computed following the logic used in Case 2. A configuration illustrating Case 3 is given Fig. 8, where the final proximal points are given by $P(C_1^b, C_2^t)$ (details are given in Sec. 4).

3 Formulation of the Proximity Functions for Distinct Pairs of Primitives

As described in Sec. 2, the original problem can be reduced to the computation of proximity of five combinations of the primitives. The approach followed to find $P(A, B)$, where A and B are any of the five primitives, in each of the cases has the following in common:

(1) A pair of generic points, $\mathbf{p}_1 \in A$ and $\mathbf{p}_2 \in B$, are described in terms of n independent geometric parameters, $\mathbf{x} = [x_1, \ldots, x_n]^\top$. The set of parameters and their number may vary from one entity to the other.
(2) The square of the distance between the two points is then obtained as function of the parameters

$$h(\mathbf{x}) = d^2(\mathbf{p}_1, \mathbf{p}_2) = (\mathbf{p}_1 - \mathbf{p}_2) \cdot (\mathbf{p}_1 - \mathbf{p}_2) \quad (11)$$

(3) The function $h(\mathbf{x})$ is then analyzed for its extreme values, over the set of valid values of $\mathbf{x}$.
(4) Upon finding the partial derivatives and setting them to zero, one obtains the following set of equations:

$$f_i = 0, \quad \text{where} \quad f_i = \frac{\partial h}{\partial x_i}, \quad i = 1, \ldots, n \quad (12)$$

(5) Equation (12) is typically nonlinear in x_i. The unknown variables x_i are eliminated sequentially to finally obtain a univariate polynomial.
(6) The roots of the univariate polynomial are then computed. The real-valued solutions are retained and substituted back into the original/intermediate equations to complete the solution(s) to Eq. (12).
(7) The value of $d(A, B)$ is computed for all real solutions of Eq. (12). The pair of points $\mathbf{p}_1^*, \mathbf{p}_2^*$ corresponding to $\min(d(A, B))$ are the closest points for the set of primitives, and following Eq. (1), the proximity is computed as $P(A, B) = ||(\mathbf{p}_1^* - \mathbf{p}_2^*)||$.

The rest of this section deals with the specific details of the pairwise proximity functions for the relevant pairs.[7]

3.1 Proximity of Two Cylindrical Surfaces, *P(S, S)*.

The following are the steps to be followed to find the shortest distance between two cylindrical surfaces:

- Without any loss of generality, the cylinder E_1 is considered to be along the $\mathbf{z}_1$ axis as shown in Fig. 6. Any point on the cylindrical surface S_1 can be represented as

$$\mathbf{p}_1 = [r_1 \cos\theta_1, r_1 \sin\theta_1, l_1 t_1]^\top$$
$$\text{where } t_1 \in [0, 1],\ \theta_1 \in [0, 2\pi]$$

- The cylinder E_2 is considered at a generic configuration given by the position of its origin, as $\mathbf{o} = [o_x, o_y, o_z]^\top$, and orientation, given in terms of the unit quaternion (see, e.g., Ref. [14]) $\mathbf{q} = [q_0, q_1, q_2, q_3]^\top$, with q_0 being the scalar part of the quaternion. Any point on S_2, with respect to the global frame of reference, is given by

$$\mathbf{p}_2 = \mathbf{o} + \mathbf{R}[r_2 \cos\theta_2, r_2 \sin\theta_2, l_2 t_2]^\top$$
$$\text{where } t_2 \in [0, 1],\ \theta_2 \in [0, 2\pi]$$

The *rotation matrix* $\mathbf{R} \in \mathbb{SO}(3)$ is obtained in terms of the unit quaternion components (see, e.g., Ref. [14], p. 16). The parameters $\mathbf{x} = [\theta_1, \theta_2, t_1, t_2]^\top$ in this case.

- The distance function is obtained as shown in Eq. (11).
- Subsequently, the equations $f_i = 0$, $i = 1, \ldots, 4$, are obtained.
- The equations $f_3 = 0$ and $f_4 = 0$ turn out to be linear in t_1, t_2. Therefore, these yield *unique* solutions for t_1 and t_2 in the closed form, in terms of θ_1 and θ_2.
- These solutions are substituted into $f_1 = 0$ and $f_2 = 0$ to obtain two new equations, $g_1(\theta_1, \theta_2) = 0$ and $g_2(\theta_1, \theta_2) = 0$, respectively. Upon further study, it is found that g_1 and g_2 can be factorized, and they share a common factor $k(\theta_1, \theta_2)$

$$\begin{aligned}
g_1 &= m(\theta_1)k(\theta_1, \theta_2)\\
g_2 &= j(\theta_2)k(\theta_1, \theta_2), \quad \text{where}\\
m(\theta_1) &= (q_0q_2 + q_1q_3)\cos\theta_1 - (q_0q_1 - q_2q_3)\sin\theta_1\\
j(\theta_2) &= (q_0q_2 - q_1q_3)\cos\theta_2 - (q_0q_1 + q_2q_3)\sin\theta_2, \quad \text{and}\\
k(\theta_1, \theta_2) &= (q_0q_1 - q_2q_3)o_x + (q_0q_2 + q_1q_3)o_y\\
&\quad + ((-q_0q_1 + q_2q_3)\cos\theta_1 - (q_0q_2 + q_1q_3)\sin\theta_1)r_1\\
&\quad + ((q_0q_1 + q_2q_3)\cos\theta_2 + (q_0q_2 - q_1q_3)\sin\theta_2)r_2
\end{aligned} \quad (13)$$

The equations $g_1 = 0$ and $g_2 = 0$ can be satisfied simultaneously in any one of the following five cases:

(a) $k \neq 0, m = 0, j = 0$
(b) $k = 0, m \neq 0, j \neq 0$
(c) $k = 0, m \neq 0, j = 0$
(d) $k = 0, m = 0, j \neq 0$
(e) $k = 0, m = 0, j = 0$

The general scenario is given by case (a)—the solution consists of only isolated points in the θ_1–θ_2 space, which are obtained in closed form. Case (b) admits only a one-dimensional solution, namely, a *curve* in the θ_1–θ_2 space that satisfies the equation $k = 0$. This curve corresponds to the

[7]For the relevant pairs discussed below, the first geometric entity belongs to the cylinder E_1, and the second to E_2. However, the mathematical formulation is generic and exhaustive, so all the possible cases are covered.

curve of intersection, when the cylinders interfere. The cases (c) and (d) admit isolated solutions, which lie on the curve $k=0$. Case (e) includes the interesting scenario where the curve of intersection shrinks to a point, and the isolated solutions obtained from $m=j=0$ coincide with this point. Thus, this case captures the special case of a single point interference between the two cylinders, as shown in Fig. 9(*a*).

In each case, the real solutions are retained and substituted back to obtain t_1 and t_2—thus completing the computation of all the unknowns.

- Among the real solutions of $\mathbf{x}$, the one corresponding to $\min(d(S_1, S_2))$ is then found out. If this solution is such that $t_1, t_2 \in [0, 1]$, then the scenario falls under case 1. If either $t_1 \notin [0,1]$ or $t_2 \notin [0,1]$, then the situation represents Case 2, and if both $t_1, t_2 \notin [0,1]$, then Case 3.

The formulation captures the special cases as well, where the solutions may be of dimension(s) one or two, or involve interference of cylinders. These are described below.

- If L_1 and L_2 are parallel, then $\mathbf{z}_1 \cdot \mathbf{z}_2 = 1$. Since $\mathbf{z}_2 = \mathbf{R}\mathbf{z}_1$ (see Fig. 9(*d*)), this means $\mathbf{z}_1^\top \mathbf{R}\mathbf{z}_1 = 1 = 1 - 2q_1^2 - 2q_2^2$. Hence, $q_1 = q_2 = 0$, and since $\mathbf{q}$ is a unit quaternion, $q_0^2 + q_3^2 = 1$. In this case, f_1 and f_2 simplify to become functions of θ_1 and θ_2 only, and are free of t_1 and t_2. Further, $f_3 = -f_4 = l_1 t_1 - l_2 t_2 - o_z$. This implies that if $o_z \in [-l_2, l_1]$, then f_3 (or f_4) can vanish for some combination of t_1, t_2, and this situation would fall under Case 1. Else, it would be considered under Case 3.
- In the antiparallel case, $\mathbf{z}_1 \cdot \mathbf{z}_2 = \mathbf{z}_1^\top \mathbf{R}\mathbf{z}_1 = -1$, and hence $q_1^2 + q_2^2 = 1$. Consequently, $q_0 = q_3 = 0$. If $o_z \in [0, l_1 + l_2]$, then this scenario falls under Case 1, else Case 3.
- If the cylinders interfere, i.e., the distance obtained is zero and both t_1, $t_2 \in [0, 1]$, then the simultaneous vanishing of

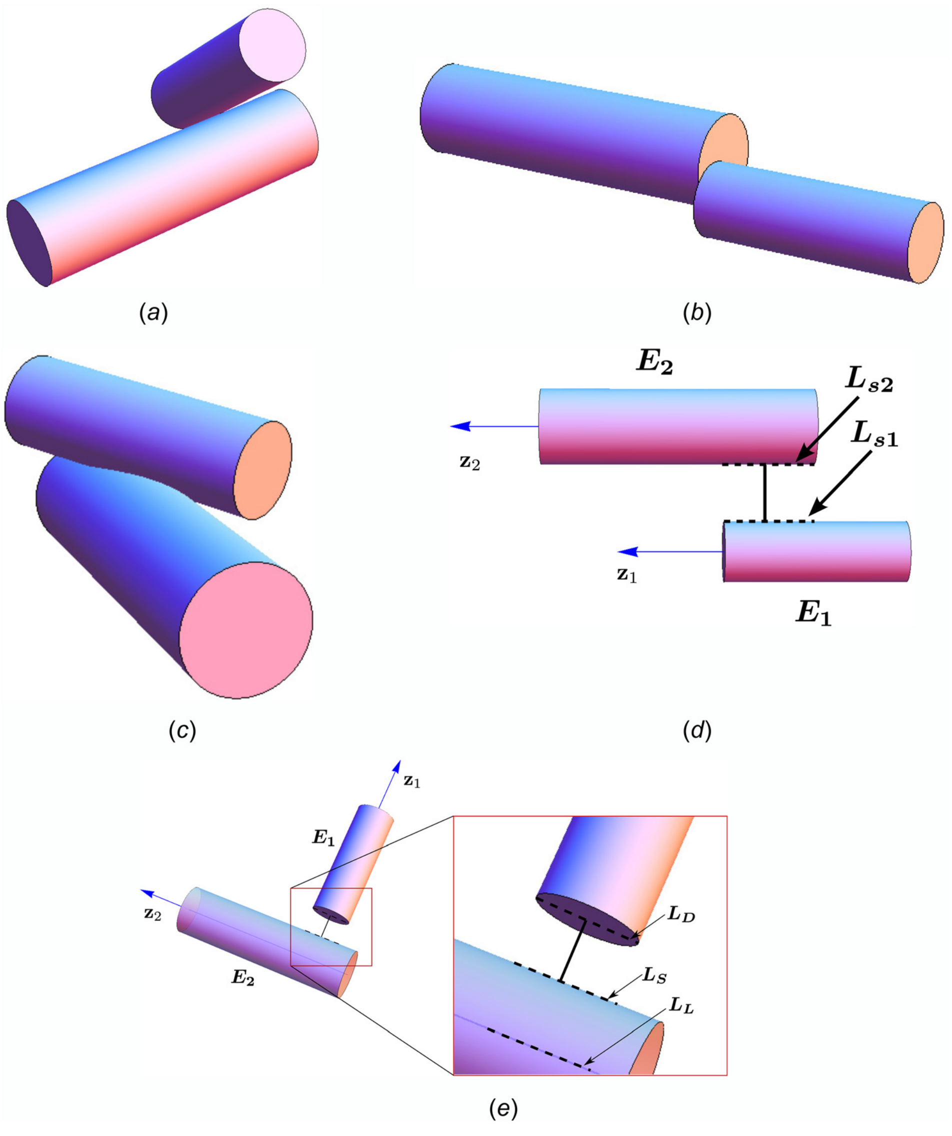

Fig. 9 Examples of special cases. These include the cases when there is interference and the solution of proximal points being one-/two-dimensional. (*a*) One point interference: the solution is given by *P*(*S*, *S*). (*b*) One common face: the two cylinders are parallel to each other and the faces overlap when seen along the axes. It has a two-dimensional solution. (*c*) Interfering cylinders: the curve of intersection is given by the formulation for finding *P*(*S*, *S*) described in Sec. 3.1. (*d*) Solution given by *P*(*S*, *S*). The axes of the cylinders are parallel to each other and the proximal points form line segments L_{s1} on the cylinder E_1 and L_{s2} on the cylinder E_2. (*e*) Solution given by *P*(*D*, *L*). The axes of the cylinders are orthogonal to each other and the proximal points form line segments L_D on the disk and L_S on cylindrical surface.

g_1, g_2 implies $k=0$. As mentioned above, the proximal points lie on the curve of intersection of the two cylinders in this case.[8]

3.2 Proximity of a Disk and a Line Segment, *P(D, L)*.

As explained in Sec. 1, the disks forming the end caps of the cylinder are parts of infinite planes. However, the part of these planes cut off by the cylinder is of interest in the following. Any point on the disk of radius r_1 may be represented as

$$\mathbf{p}_1 = [r_1 s_1 \cos\theta_1, r_1 s_1 \sin\theta_1, 0]^\top, \quad \text{where} \\ s_1 \in [0,\ 1),\ \theta_1 \in [0,\ 2\pi] \tag{14}$$

The disk is bounded by the circle corresponding to $s_1=1$. The shortest distance between a disk and a line segment can be obtained in the following manner:

- The line segment is considered at a generic configuration given by $\mathbf{o}$ and $\mathbf{R}$, such that any point on it is given by $\mathbf{p}_2=\mathbf{o}+\mathbf{R}[0,\ 0,\ l_2t_2]^\top$, where $t_2 \in [0,\ 1]$ and l_2 is the length of the line segment under consideration.
- Similar to the case of $P(S,\ S)$, a distance function $h(\mathbf{x})$ is formulated and its partial derivatives with respect to $\mathbf{x}=[s_1,\ \theta_1,\ t_2]^\top$ are set to zero to obtain the equations $f_i=0$, $i=1,\ 2,\ 3$.
- The equation $f_3=0$ turns out to be linear in t_2, and therefore, yields a solution for t_2 readily as a function of θ_1 and s_1.
- Upon substituting the expression of t_2 in $f_1=0$ and $f_2=0$, new equations $g_1=0$ and $g_2=0$ are obtained, both of which are linear in s_1.
- The expression for s_1 in terms of θ_1 is obtained from $g_1=0$. Upon substituting this expression in $g_2=0$ and simplifying, one obtains $g_3=0$, which is linear in $\cos\theta_1$ and $\sin\theta_1$.
- The two solutions of θ_1 can be obtained analytically by solving $g_3=0$. Upon back substitution, s_1 and t_2 are obtained.
- If $s_1 \notin [0, 1)$, then the proximal point $\mathbf{p}_1^*$ falls beyond the finite disk under consideration. This implies $\mathbf{p}_1^*$ would finally lie on the boundary circle of the disk, and the shortest distance between the bounding circle C_1 and the line $\mathbf{z}_2$ is computed by $P(C,\ L)$ function. The scenario is depicted in Fig. 7(*a*).

The formulation captures the special case where the normal to the disk, i.e., $\mathbf{z}_1$, is perpendicular to the line segment L_2 (see Fig. 9(*e*)). This implies that $\mathbf{z}_1\cdot\ \mathbf{z}_2=0$. Consequently, $\mathbf{z}_1^\top\mathbf{R}\mathbf{z}_1 = 0 = 1-2q_1^2-2q_2^2$, and hence, $q_1^2+q_2^2=q_0^2+q_3^2 = 1/2$. Equivalently, $q_0 = a\cos\phi_1$, $q_3 = a\sin\phi_1$, $q_1 = a\cos\phi_2$, and $q_2 = a\sin\phi_2$, where $a = 1/\sqrt{2}$ and ϕ_1, ϕ_2 are constants. In this case, g_1 and g_2 factorize as

$$\begin{aligned} g_1(\theta_1, s_1) &= g_4(\theta_1)\cos(\theta_1-\phi) \\ g_2(\theta_1, s_1) &= g_4(\theta_1)\sin(\theta_1-\phi), \quad \text{where} \quad \phi=\phi_1+\phi_2 \\ g_4(\theta_1, s_1) &= -o_x\cos\phi - o_y\sin\phi + r_1s_1\cos(\theta_1-\phi) \end{aligned} \tag{15}$$

The general solution is obtained when $g_1^2+g_2^2=g_4^2=0$. Thus, the solution lies on a curve in the space θ_1–s_1, satisfying $g_4(\theta_1,\ s_1)=0$. It represents a line (see Fig. 9(*e*)). A special solution emerges when $\cos(\theta_1-\phi)=0$, provided $o_x/o_y = -\tan\phi = [(q_3q_1)+(q_0q_2)]/[(q_3q_2)-(q_0q_1)]$. The solution set is given by the two values of θ_1 satisfying $\cos(\theta_1-\phi)=0$ and all $s_1 \in [0,\ 1)$. Similarly, solutions in the form of isolated points are obtained when $\sin(\theta_1-\phi)$ vanishes.

3.3 Proximity of a Circle and a Line Segment, *P(C, L)*.

The following steps are followed to find the shortest distance between a circle and a line segment:

- As the circle under consideration forms the boundary of the disk, the distance function is the same as in $P(D,\ L)$, except that in this case, $s_1=1$. The proximity conditions, $f_i=0$, $i=1,\ 2$, are obtained in terms of $\mathbf{x}=[\theta_1,\ t_2]^\top$.
- Upon solving linearly for t_2 from $f_2=0$, one obtains

$$\begin{aligned} t_2 = &-(o_z(q_0^2-q_1^2-q_2^2+q_3^2)+2r_1\sin\theta_1(q_0q_1-q_2q_3) \\ &-2r_1\cos\theta_1(q_0q_2+q_1q_3)+(2q_0q_2+2q_1q_3)o_x \\ &+(2q_2q_3-2q_0q_1)o_y)/l_2 \end{aligned} \tag{16}$$

Substituting the value of t_2 into $f_1=0$ leads to $g_1=0$. Since g_1 is nonlinear in the sine and cosine of θ_2 in this case, the trigonometric entities in g_1 are converted to algebraic entities using the standard half-tangent substitution (see, e.g., Ref. [15]), to obtain a quartic polynomial, g_2, in v, where $v=\tan(\theta_2/2)$.

- The equation $g_2=0$ can be solved analytically to obtain v and θ_2 therefrom. Upon substituting back into Eq. (16), t_2 is obtained.

3.4 Proximity of a Disk and a Circle, *P(D, C)*.

The steps followed to find the shortest distance between a disk and a circle are given below:

- A point $\mathbf{p}_1$ on the disk is represented as in Eq. (14). The circle of radius r_2 is considered in a generic configuration given by $\mathbf{o}$ and $\mathbf{R}$, such that any point on it is $\mathbf{p}_2=\mathbf{o}+\mathbf{R}[r_2\cos\theta_2, r_2\sin\theta_2, 0]^\top$, where $\theta_2 \in [0, 2\pi]$.
- The geometric variables in this case are $\mathbf{x}=[\theta_1,\ \theta_2,\ s_1]^\top$. Setting the partial derivatives, $\partial h/\partial x_i$, to zero, the equations $f_i=0$, $i=1,\ 2,\ 3$, are obtained.
- The equation $f_3=0$ is linear in s_1 and is solved for the same to obtain

$$\begin{aligned} s_1 = &(r_2(q_0^2-q_3^2)\cos(\theta_1-\theta_2)+2q_0q_3r_2\sin(\theta_1-\theta_2) \\ &+q_1^2r_2\cos(\theta_1+\theta_2)+2q_1q_2r_2\sin(\theta_1+\theta_2) \\ &-q_2^2r_2\cos(\theta_1+\theta_2)+o_x\cos\theta_1+o_y\sin\theta_1)/r_1 \end{aligned} \tag{17}$$

The expression for s_1 is then substituted into $f_1=0$ and $f_2=0$ to obtain $g_1=0$ and $g_2=0$, respectively.

- The trigonometric entities in g_1 and g_2 are converted to algebraic entities with the help of half-tangent substitutions to obtain $g_3=0$ and $g_4=0$ in terms of v_1, v_2, where $v_1=\tan(\theta_1/2)$ and $v_2=\tan(\theta_2/2)$.
- The *resultant* (see, e.g., Ref. [16], pp. 77–94) of the equations $g_3(v_1,\ v_2)=0$ and $g_4(v_1,\ v_2)=0$ with respect to v_2 is computed, to obtain $g_5(v_1)=0$, an eight-degree polynomial equation in v_1. The polynomial g_5 factorizes into two quartics: $g_6(v_1)=0$ and $g_7(v_1)=0$. These quartics can be solved analytically to obtain their roots.
- The real solutions to $g_5=0$ are retained, and upon back substitution into $g_3=0$ and $g_4=0$, solutions to v_2 are obtained. The angles θ_1 and θ_2 are obtained from the real values of v_1 and v_2, respectively, by using tangent half-angle relationship, which upon substituting in Eq. (17) gives s_1. If $s_1 \notin [0, 1)$, then the boundary circle of the disk is considered and the shortest distance between the two circles is computed using $P(C, C)$.

The formulation also brings out the special case when the plane of the circle is parallel to the plane of the disk, i.e., $\mathbf{z}_1\cdot\ \mathbf{z}_2=\pm1$. As explained in Sec. 3.1, when $\mathbf{z}_1\cdot\mathbf{z}_2=1$, $q_1=q_2=0$, and consequently, $q_0^2+q_3^2=1$. The condition $q_0^2+q_3^2=1$ can be restated as $q_0=\cos\phi$ and $q_3=\sin\phi$, where ϕ is a constant. Under this condition, g_4 factorizes as

$$\begin{aligned} g_4 = &2g_3(v_1,v_2)((1+v_1+(v_1-1)v_2)\cos\phi \\ &-(1-v_1+(v_1+1)v_2)\sin\phi) \\ &((1-v_1+(v_1+1)v_2)\cos\phi \\ &+(1+v_1+(v_1-1)v_2)\sin\phi) \end{aligned} \tag{18}$$

[8]It may be noted that the analytical description of the curve can be used for the development of the surfaces of the intersecting cylinders.

Table 1 Configuration of the cylinders and the corresponding pairs of points having the least distance between them (Figs. 6–8)

Figure	$\mathbf{q}$	$\mathbf{o}$	$\mathbf{p}_1^*, \mathbf{p}_2^*$
6	$\left[-\frac{1}{5}, \frac{3}{10}, \frac{1}{\sqrt{2}}, \frac{\sqrt{37}}{10}\right]^\top$	$[8, -8, 8]^\top$	$[1.99, -0.17, 6.66]^\top, [6.12, -0.51, 6.66]^\top$
7(*a*)	$\left[0, \frac{2}{5}, -\frac{3}{10}, \frac{\sqrt{3}}{2}\right]^\top$	$[6, -6, 8]^\top$	$[5.31, -4.87, 10.12]^\top, [1.47, -1.35, 10.12]^\top$
7(*b*)	$\left[0, \frac{2}{5}, -\frac{3}{10}, \frac{\sqrt{3}}{2}\right]^\top$	$[6, -6, 12]^\top$	$[1.53, -1.29, 12.00]^\top, [5.53, -4.64, 14.05]^\top$
7(*c*)	$\left[\frac{3}{10}, \frac{2}{5}, \frac{\sqrt{3}}{2}, 0\right]^\top$	$\left[\frac{-7}{10}, 1, -7\right]^\top$	$[1.16, 0.14, 0]^\top, [1.06, 0.14, -5.57]^\top$
8	$\left[\frac{3}{10}, \frac{2}{5}, \frac{-3}{10}, \frac{\sqrt{33}}{5\sqrt{2}}\right]^\top$	$[-12, 12, -18]^\top$	$[-1.99, 0.14, 0]^\top, [-3.69, 0.25, -6.90]^\top$

Note: The proximity can be calculated as $\|\mathbf{p}_1^* - \mathbf{p}_2^*\|$. The proximal points $\mathbf{p}_1^*, \mathbf{p}_2^*$ mentioned in the table are rounded-off to two places after the decimal point for the sake of brevity. However, the exact results in two sample cases are reproduced below to demonstrate the capability of the proposed method in obtaining closed-form solutions. The proximity in Fig. 6 is $8\sqrt{\frac{990\sqrt{2}}{2419} - \frac{4\sqrt{37}}{59} + 1} - \frac{9}{2}$. In Fig. 7(*a*), the proximity is $\frac{\sqrt{69365 - \sqrt{3}c_7 - c_8}}{20\sqrt{2}} - 2$, where $c_1 = 4{,}720{,}590{,}918{,}830{,}961{,}665{,}569$, $c_2 = 18{,}672{,}176{,}766{,}028{,}800$, $c_3 = 104{,}197{,}414{,}989$, $c_4 = 23{,}462{,}497$, $c_5 = 241{,}269{,}141{,}468{,}479$, $c_6 = \sqrt[3]{c_1 + c_2\sqrt{c_3}}$, $c_7 = \sqrt{c_4 - \frac{c_5}{c_6} + c_6}$, and $c_8 = \sqrt{6c_4 + \frac{3c_5}{c_6} - 3c_6 + \frac{687075125760\sqrt{3}}{c_7}}$.

The factor $g_3(v_1, v_2)$ is linear in v_1, v_2. All the real pairs of v_1, v_2 satisfying $g_3 = 0$ and the corresponding θ_1 and θ_2 define a one-dimensional solution, signifying the projection of the circle onto the disk. The other factors of g_4 lead to isolated points, which may or may not fall on the one-dimensional solution mentioned above. The conditions for the case $\mathbf{z}_1 \cdot \mathbf{z}_2 = -1$ can be derived similarly.

3.5 Proximity of Two Circles, *P*(*C*, *C*). The steps to be followed to find the shortest distance between two circles are delineated below:

- The distance function in this case is the same as that of $P(D, C)$, as discussed in Sec. 3.4, with $s_1 = 1$.
- Equations $f_1 = 0$ and $f_2 = 0$ are obtained next.
- The equations $f_1 = 0$ and $f_2 = 0$ are treated similar to $g_1 = 0$ and $g_2 = 0$ as in the case of $P(D, C)$ (Sec. 3.4), to obtain a univariate polynomial, g_3, of degree eight in terms of v_1, where $v_1 = \tan(\theta_1/2)$. In this case, however, the polynomial does not factorize into lower degree polynomials, and hence needs to be solved numerically.

All the coefficients of the polynomial g_3 have been obtained in closed form, using symbolic computations in the CAS MATHEMATICA. The *size* of the polynomial[9] g_3 is about 700 kB in MATHEMATICA. In Sec. 4, the coefficients of the polynomial are given for a numerical example.

4 Numerical Studies

In this section, the algorithm developed in Secs. 2 and 3 is applied to various configurations of a pair of cylinders to illustrate the different cases and the corresponding distance queries. The numerical parameters used are: $r_1 = 2$, $r_2 = 5/2$, $l_1 = 12$, and $l_2 = 18$. As the numerical values chosen are solely for the purpose of illustration, they are considered to be free of any units in this paper. Figures 6–8 show five different configurations of the pair of cylinders and the corresponding proximal points. Table 1 lists the relative position $\mathbf{o}$ and orientation $\mathbf{q}$, and the corresponding proximity of the cylinders, for each of the cases.

The scenario depicted in Fig. 8 represents the worst case from a computational perspective, as it involves the longest sequence of steps before ending up at the proximity result. For the sake of clarity, this example is explained in detail in the following. The analytical form of the expressions is not included hereafter for the sake of brevity.

(1) First, treating the problem as in case 1, using $P(S_1, S_2)$, the corresponding values of θ_1, θ_2, t_1, and t_2 are computed. The value of $d(S_1, S_2)$ is minimum for $[\theta_1, \theta_2, t_1, t_2] = [-2.568, -0.968, -0.514, 1.127]$. As $t_1, t_2 \notin [0, 1]$, this results in the classification Case 3 (see Sec. 2).

(2) The disks to be considered in this example would correspond to $t_1 = 0$ and $t_2 = 1$, i.e., D_1^b and D_2^t, as they are closest to the proximal points, $\mathbf{p}_1^*$ and $\mathbf{p}_2^*$, obtained from $P(S_1, S_2)$.

(3) The problem is now equivalent to first finding $P(D_1^b, S_2)$ and then $P(D_2^t, S_1)$, as if solving for two problems of the type Case 2.

(4) Following the algorithm given in Fig. 5, $P(D_1^b, z_2)$ is computed to obtain $[\theta_1, s_1, t_2] = [-1.237, 7.508, 2.000]$. As $s_1 > 1$, the proximity is queried between C_1^b and z_2 using $P(C_1^b, z_2)$. The quartic equation $g_2 = 0$ is obtained next: $g_2 = 4.315v_1^4 + 0.804v_1^3 + 4.102v_1^2 + 3.271v_1 - 5.683$. The real roots obtained are $[\theta_1, t_2] = [-1.631, 1.376]$ or $[1.227, 1.240]$.The solution corresponding to $\min(d(C_1^b, z_2))$ is $\theta_1 = -1.631$, $t_2 = 1.376$. Since $t_2 > 1$, this scenario belongs to Case 2.2.

(5) Under Case 2.2, the shortest distance between D_2^t (since $t_2 > 1$) and D_1^b is found. To do this, first $P(C_1^b, D_2^t)$ is computed, following the algorithm depicted in Fig. 5. The polynomials g_6 and g_7 are obtained as

$$\begin{aligned} g_6 = &-5.663v_1^4 + 24.432v_1^3 - 14.709v_1^2 \\ &-24.432v_1 - 5.663 \end{aligned}$$

$$\begin{aligned} g_7 = &-54.324v_1^4 + 143.061v_1^3 - 139.528v_1^2 \\ &-143.061v_1 - 54.324 \end{aligned}$$

The real solutions obtained are $[s_1, \theta_1, \theta_2] = [3.611, -0.696, 2.144]$ or $[2.894, -0.806, -0.997]$. As $s_1 \notin [0, 1)$ in any of the solutions, the shortest distance between C_1^b and C_2^t is found following the logical sequence delineated in Sec. 3.5. The polynomial g_3 is given by $g_3 = -38.654v_1^8 + 1004.620v_1^7 + 3572.320v_1^6 + 2542.520v_1^5 + 6863.510v_1^4 + 1172.350v_1^3 +$

[9]Size of the polynomial, in this context, is the amount of computer memory used to represent the expression in a CAS' internal format.

Table 2 Configuration of the cylinders for the special cases

Figure	$\mathbf{q}$	$\mathbf{o}$
9(*a*)	$\left[\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 0\right]^\top$	$\left[-4, \frac{9}{2}, 4\right]^\top$
9(*b*)	$[0, 0, 0, 1]^\top$	$[2, 0, 12]^\top$
9(*c*)	$\left[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{\sqrt{22}}{5}\right]^\top$	$[-4, 2, 2]^\top$
9(*d*)	$[0, 0, 0, 1]^\top$	$[2, 8, 6]^\top$
9(*e*)	$\left[\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 0\right]^\top$	$[-4, 0, -5]^\top$

Table 3 The time taken by the algorithm for different cases (averaged over a total of 1×10^6 randomized trials)

Case	Average time taken per trial (in μs)
1	0.75
2.1	4.69
2.2	54.45
3	72.28

$3171.970 v_1^2 - 846.209 v_1 - 114.347$. The real results obtained are $[\theta_1, \theta_2] = [-2.515, 2.638]$, $[-0.197, -0.939]$, $[0.572, 2.199]$, $[3.073, -0.534]$. The solution corresponding to $\min(d(C_1^b, C_2^t))$ is $\theta_1 = 3.073$ and $\theta_2 = -0.534$. The corresponding shortest distance is 7.106 and the proximal pair of points obtained are $\mathbf{p}_1^* = [-1.995, 0.138, 0]^\top$ on E_1 and $\mathbf{p}_2^* = [-3.691, 0.253, -6.899]^\top$ on E_2, as shown in Fig. 8. Following this, the shortest distance between D_1^b and C_2^t is found leading to $P(C_1^b, C_2^t)$, which has been already found and need not be computed again.

(6) As mentioned in step 3, the steps are repeated to find $P(D_2^t, S_1)$. This leads to $P(D_1^b, D_2^t)$ again, and hence need not be computed.

(7) Therefore, the proximity of the cylinders is 7.106.

In addition to the cases shown in Figs. 6–8, the analytical approach described in this paper can also be applied to the special cases with one-dimensional or two-dimensional solutions to the problem of the shortest distance, such as those shown in Fig. 9. Instead of synthesizing the geometric conditions for these cases, the analytical formulation directly leads to the solutions in every possible case—a fact that forms an important advantage of this method of solving the problem. The configuration shown in Fig. 9(*d*) results in a one-dimensional solution, i.e., namely, a pair of line segments. The results show the shortest distance as zero for other configurations and lead to the point, curve, or surface of intersection, as the case may be. Table 2 shows the $\mathbf{q}$ and $\mathbf{o}$ corresponding to the cases depicted in Fig. 9.

For example, in Fig. 9(*c*), the intersection curve can be obtained directly from Eq. (13) as $k(\theta_1, \theta_2) = 1.046 + 0.295\cos\theta_1 + 0.569\cos\theta_2 - 0.455\sin\theta_1 - 0.369\sin\theta_2$.

Similarly, in Fig. 9(*e*), the one-dimensional solution can be readily obtained from Eq. (15) as $g_4 = s_1 \sin\theta_1$, which upon substitution into $f_1 = 0$ and $f_2 = 0$ yields $f_1 = -f_2 = 2 + s_1 - 9t_2$. This represents a one-dimensional solution, as shown by the dashed lines in the figure.

5 Numerical Validation Via Randomized Checks

In order to validate the algorithm, two cylinders are considered. One of the cylinders is fixed with radius $r_1 = 2$ units and length $l_1 = 5$ units. The radius r_2 and the length l_2 of the second cylinder are chosen randomly such that $0 < r_2 \leq 10 r_1$ and $0 < l_2 \leq 10 l_1$. The position and orientation of the second cylinder, with respect to the fixed cylinder, are also chosen randomly. The origin of the axis system of the second cylinder is then chosen randomly as $0 < \|\mathbf{o}\| \leq 5 l_1$ such that all the general cases are covered. The pose of the second cylinder is thereafter found in terms of unit quaternions and position of its "origin" $\mathbf{o}$.

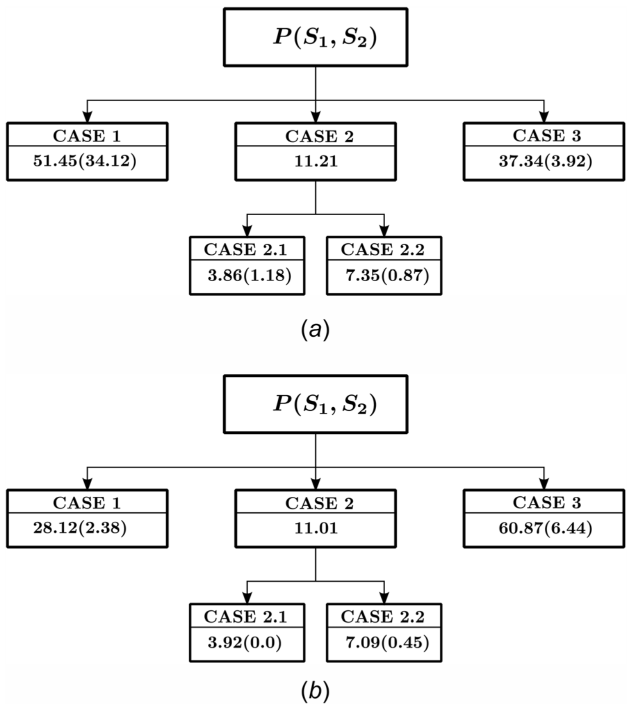

Fig. 10 Summary of numerical experiments depicting actual occurrences of each case and interference at each level: (*a*) near field and (*b*) far field. The values, e.g., "51.45(34.12)" in Case 1 for near field, denote that the observed occurrences of the particular case are 51.45% and actual interference detected is 34.12%.

An application is created in AUTODESK INVENTOR[10] to validate the results. A total of 100,000 such random scenarios are generated, and the results are verified using this application. It is to be noted that the random checks do not fully cover the scope of the algorithm as the probability of encountering any of the special cases is exactly zero, since they are only finite in number. Nevertheless, the results of the random checks validate the proposed algorithm for the general cases and provide an estimate of the computational time[11] required for the algorithm to compute the distances and proximal points, as given in Table 3. The algorithm is coded in C++ programming language. It is imperative to know how the algorithm performs in different real-life application scenarios. To study this, the random trials are further divided into two categories, *near field* and *far field*, based on the relative distance of one cylinder from the other, given by $\|\mathbf{o}\|$. The *near field* is described as the situation when the cylinders are "close to" each other. It is designated as $\|\mathbf{o}\| \leq \max((l_1 + l_2), (r_1 + r_2))$. Similarly, *far field* is described as the situation when the cylinders are "far" from each other. It is given by $\|\mathbf{o}\| > \max((l_1 + l_2), (r_1 + r_2))$. The key difference between the two fields lies in the fact that the chances of collision increase for *near field*, and hence, probability of detection in first level is increased, thereby reducing the overall computations. This can be observed in the data given in Fig. 10, where 51.45% of the trials fall under Case 1 for near field as opposed to 28.12% for far field.

[10]Autodesk® Inventor LT™, version 2014: http://www.autodesk.com/products/inventor-lt-family/overview

[11]The computations were performed on a PC with 32 GB of RAM and Intel® Core™ i7-4770 CPU running at 3.40 GHz. The compiler used was gcc, version 4.8.4.

Such an analysis is applicable in many fields, e.g., in hyper-redundant robots and modeling of cables, wherein the number of cylinders is large and an estimate of the computational time required to find proximity and the corresponding points is necessary.

6 Conclusion

A novel analytical formulation for computing the proximity of two arbitrary cylinders in space has been presented in this paper. The proximity of the cylinders and the corresponding pair of points on the cylinders are obtained by a hierarchical computation of the proximity between five pairs of geometric primitives, which make up each cylinder. The formulation of the problem is strictly analytical in nature, and the results are obtained in closed form in all the cases, except in one where the analytical work is limited to the computation of the coefficients of an eight-degree polynomial, whose solutions must be obtained by numerical means.

Owing to the analytical nature of the formulation, and the use of unit quaternions to represent relative orientations, two major advantages are derived: all the general as well as special cases are identified exactly, and the numerical implementation is very fast, as most of the work is done in the symbolic precomputations stage. The C++ implementation used to generate all the results in this paper finds the proximity of 20,000 arbitrary pairs of cylinders in just one CPU-second on a PC running at 3.40 GHz. The results obtained are verified in each case by means of comparison with a general-purpose commercial software, AUTODESK INVENTOR. As expected, due to the specialized nature of the algorithm, it is about 1000 times faster than the general-purpose software, while being equally accurate. The proposed algorithm is expected to have significant applications in robotics, in particular in the field of path planning.

Acknowledgment

The authors would like to thank Mr. Rajeevlochana G. Chittawadigi for some useful discussions, and for providing a few references at the initial stages of this research. He also provided initial guidance to the first author toward the development of the application in Autodesk Inventor that has been used to verify the results obtained.

References

[1] Ketchel, J., and Larochelle, P., 2006, "Collision Detection of Cylindrical Rigid Bodies for Motion Planning," IEEE International Conference on Robotics and Automation, Orlando, FL, May 15–19, pp. 1530–1535.
[2] ElMaraghy, W., Valluri, S., Skubnik, B., and Surry, P., 1994, "Intersection Volumes and Surface Areas of Cylinders for Geometrical Modeling and Tolerancing," Comput.-Aided Des., **26**(1), pp. 29–45.
[3] Tang, T. D., 2014, "Algorithms for Collision Detection and Avoidance for Five-Axis NC Machining: A State of the Art Review," Comput.-Aided Design, **51**, pp. 1–17.
[4] Biermann, D., Joliet, R., and Michelitsch, T., 2008, "Fast Distance Computation Between Cylinders for the Design of Mold Temperature Control Systems," *Advances in Computational Intelligence—Theory and Practice*, Reihe CI-259/08, SFB 531, Technical University of Dortmund, Dortmund, Germany.
[5] Hudgens, J., and Arai, T., 1993, "Planning Link-Interference-Free Trajectories for a Parallel Link Manipulator," IECON'93, Maui, HI, Nov. 15–19, Vol. 3, pp. 1506–1511.
[6] Merlet, J.-P., 1995, "Determination of the Orientation Workspace of Parallel Manipulators," J. Intell. Rob. Syst., **13**(2), pp. 143–160.
[7] Patel, R. V., Shadpey, F., Ranjbaran, F., and Angeles, J., 2005, "A Collision-Avoidance Scheme for Redundant Manipulators: Theory and Experiments," J. Field Rob., **22**(12), pp. 737–757.
[8] Chittawadigi, R., and Saha, S., 2013, "An Analytical Method to Detect Collision Between Cylinders Using Dual Number Algebra," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Nov. 3–7, pp. 5353–5358.
[9] Srivatsan, R. A., and Bandyopadhyay, S., 2013, "An Analytical Formulation for Finding the Proximity of Two Arbitrary Cylinders in Space," Conference on Advances in Robotics, AIR'13, ACM, New York, p. 48.
[10] Vranek, D., 2002, "Fast and Accurate Circle-Circle and Circle-Line 3D Distance Computation," J. Graphics Tools, **7**(1), pp. 23–31.
[11] Karnam, M. K., Srivatsan, R. A., and Bandyopadhyay, S., "Computation of the Safe Working Zone of Parallel Manipulators," Mech. Mach. Theory (in press).
[12] Sreenivasan, S., Goel, P., and Ghosal, A., 2010, "A Real-Time Algorithm for Simulation of Flexible Objects and Hyper-Redundant Manipulators," Mech. Mach. Theory, **45**(3), pp. 454–466.
[13] Chirikjian, G., and Burdick, J., 1994, "A Modal Approach to Hyper-Redundant Manipulator Kinematics," IEEE Trans. Rob. Autom., **10**(3), pp. 343–354.
[14] Selig, J., 2005, *Geometric Fundamentals of Robotics* (Monographs in Computer Science), Springer-Verlag, New York.
[15] Ghosal, A., 2009, *Robotics: Fundamental Concepts and Analysis*, 4th ed., Oxford University Press, New Delhi, India.
[16] Cox, D., Little, J., and O'Shea, D., 1991, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Springer-Verlag, New York.