Fast Distance Computation between Cylinders for the Design of Mold Temperature Control Systems

D. Biermann, R. Joliet, T. Michelitsch

Institute of Machining Technology (ISF),
Technische Universität Dortmund, Germany

Optimization algorithms for the design of mold temperature control systems based on deep-hole bores have to assure that the minimal distances between the bores meet given safety margins. If the bores are geometrically modeled as cylinders, this leads to the necessity of determining the minimal Euclidean distances between cylinders and testing them against the corresponding margins. In this paper a very fast and reliable algorithm for the distance computation between cylinders is introduced, which has been developed due to the run-time requirements of the problem at hand.

1 Introduction

Generating and optimizing solutions for applied problems is often a difficult task if complex objective functions and multiple constraints have to be considered. In these cases stochastic optimization algorithms such as evolutionary algorithms (EA), particle swarm optimization (PSO), and simulated annealing (SA) are commonly used. During the application of those algorithms, typically large amounts of different solutions have to be evaluated. Therefore, increasing the speed of the utilized evaluation functions is often crucial for the practical suitability of stochastic optimization algorithms.

In this paper an applied problem from the field of manufacturing engineering, the design of temperature control systems of permanent molds, is discussed in section 2. The optimization system which was developed for this problem, utilizes evolutionary algorithms [3]. During a typical optimization run of this system, millions of computations of the minimal Euclidean distance between two cylinders have to be performed.

Due to a lack of a fast, accurate, and reliable algorithm for the distance computation between cylinders, the algorithm presented in this paper was developed. It is based on the algorithm introduced by Vranek [8] in 2002. An introduction of the problem of computing the minimal distance between cylinders as well as to Vranek's algorithm is given in section 3. The advanced algorithm is presented in section 4. In addition, the accuracy and reliability tests performed as well as the special adaptations made for the applied problem at hand are discussed.

2 Design of Mold Temperature Control Systems

In many bulk production processes like injection molding and die casting, it is crucial to achieve the desired process temperatures. Controlling the temperature of permanent molds is required to obtain high product qualities and in order to reduce the production costs since the achievable cycle times depend on the mold temperatures. For this, the permanent molds are penetrated by a set of interconnected deep-hole bores through which a cooling fluid is led. Such sets of deep-hole bores are named mold temperature control systems (MTCS).

Layouting such mold temperature control systems is a challenging task since multiple objectives as well as several geometric constraints have to be considered. The objectives can be classified into thermal objectives, e. g., obtaining an intensive and homogeneous cooling of the mold, and objectives regarding the production of the MTCS, like the production costs and the time required for the manufacturing process.

In order to support the designer, an optimization system for mold temperature control systems has been developed [3]. It utilizes single- and multi-objective evolutionary algorithms such as evolution strategies [7] and the SMS-EMOA [1]. It allows optimizing given designs of MTCS as well as generating completely new designs. The optimization system utilizes different objective and penalty functions to evaluate mold temperature control systems. On the one hand the functions must be able to cope with arbitrary MTCS, on the other hand, their execution times must be as low as possible since several thousands of MTCS are evaluated during an optimization run.

The execution times of the penalty functions have a great influence on the overall run time. In the following, a focus is put on the penalty function for collisions between bores of the MTCS. In the geometric model utilized by this penalty function, the bores of the MTCS are represented as cylinders. To avoid that two nonadjacent bores do interpenetrate each other and to ensure that they meet a given safety margin, the Euclidean minimal distances between the cylinders have to be determined, which are just named 'distances' for short in the following. If a distance is larger than the given safety margin, no penalty will be assigned. Otherwise, a penalty value, which is a function of the distance, is calculated. Therefore, in the latter case the exact distance has to be known.

The algorithm for the distance computation between cylinders presented in this paper was designed with respect to the requirements of the problem at hand. It was created to be fast, in particular for typical data of this application, but at the same time it was made to be completely reliable so that it can deal with arbitrary cylinders and meet the accuracy demands of any given case.

3 Distance Computation between Cylinders

Determinating the distance between solid cylinders is possible by considering only the surfaces of the cylinders. This, however, requires an extra detection of the intersection of the surfaces. Additionally, one cylinder can completely contain the other. In both cases the distance is zero.

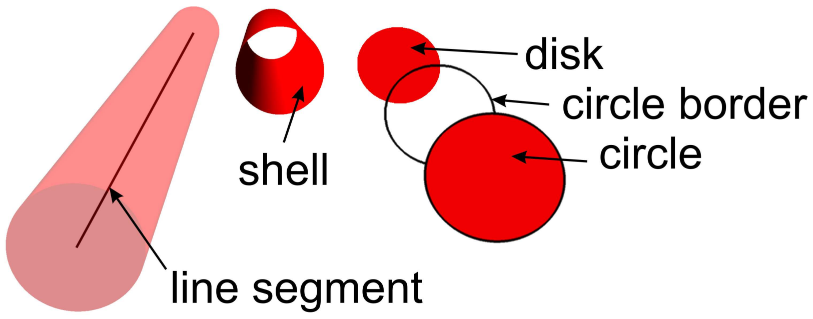

Figure 1: Simple geometric elements of cylinders regarded for the distance computation

The surface of a cylinder can be described as a quadric surface, which is trimmed by two planes. Additionally, the ends of the cylinder must be closed. The complete surface can then be partitioned into a shell and two circles, where the latter each consist of a circle border and a disk (see figure 1). In the following, only right and circular cylinders are considered, which simplifies the following steps, but is sufficient to solve the previously described problem of finding the distance between bores.

According to Lennerz and Schomer [4], the distance between two quadric surfaces is in general difficult to determine and can be computed by solving polynomials of high degree. For all quadrics with polynomials of degree five or higher, the distance cannot be computed algebraically. Finding roots for polynomials of degree three and four algebraically is possible, but due to numerical reasons not advisable. Lennerz and Schomer describe a general solution and also present the idea of decomposing the surface of the quadrics into simple shapes.

David Vranek [8] identifies four well-defined cases for the distance computation between two cylinder surfaces. For each of the cases, the distance between certain parts of the surfaces is computed, and the minimum of these distance values is the real distance between the cylinder surfaces. All the computations can be broken down into distance computations between basic geometric elements, namely line segments, circles, and disks.

Vranek presents an algorithm for the two most complicated cases, the distance computations between a circle and a line as well as between two circles. The distance computations are improvements of existing algorithms having been presented in [6]. An implementation as well as the appendix of Vranek's paper, describing and visualizing the distance cases, is available at [2].

The algorithms introduced by Vranek are based on numerical and algebraic considerations, increasing the accuracy and computational speed in relation to previous implementations and techniques. The approach for the computation of the minimum distance between two circles in three dimensions uses a direct minimization of the distance function. Thus, other extrema can be found directly by deflating a fourth degree polynomial and solving the remaining polynomial of degree two. Brent's method is used for the numerical minimum search [5]. The task of finding the minimum distance for a line and a circle in three dimensions has been solved in a similar way.

Exhaustive experiments with Vranek's code have been performed and revealed some inaccuracies. First of all for some parameter values, the distance computation between a circle and a line results in numerical inaccuracies, which make the code unstable. For the given test cases, however, the results have been correct by chance, but generally the computation may be inaccurate. The reason for not detecting this errors is based on asserting statements that are commented out. Another problem occurred while trying to compute the minimum distance between circles and lines with the circles radius being smaller than one. By normalizing the used length this error could be fixed. All in all, the tests showed that the given code performs very well with these small fixes. Finally the test suite, given by Vranek, was executed again and showed accurate results for all test cases.

Experiences with the existing algorithms show the necessity for exhaustive tests. Therefore, an extensive test suite has been build up and the newly invented and implemented methods which are, presented in this paper, were applied. The testing datasets consist of random values, real application results, and manually arranged test cases. Random tests create many different and unexpected constellations, which cover a wide range of possible input data. In contrast to this, real problem results consist mostly of discrete values, which cannot be attained by random tests. Therefore, the algorithms were validated for ordinary and frequent constellations. Finally, manually arranged tests cases were created and contain almost all constellations that might be numerically difficult to compute. The latter shall help to detect problems with rare cases, which can barely be found by chance. Each test of each class of the test suite was used as starting value for a series of individual tests. The distance for each given constellation was computed and, in the following, the parameters were transformed by increasing random values.

As previously described, the basic idea is to compute all distances for all subshapes and to use the minimal value. If all subshapes are considered, the minimum of these gives the global minimum distance between both cylinders. The consideration of surface geometry in contrast to solid cylinders must be considered separately but leads to simple tests cases, e. g., the cylinder shell can be regarded as a line. This, however, involves the implementation of additional test cases to exclude the intersection of cylinder surfaces or the case if one cylinder is located completely in the other (distance is zero).

In the following, a detailed description of the adaptation of Vranek's idea to an optimized distance computing algorithm between cylinders is presented. On the one hand, application-specific optimizations are described and, on the other hand, general purpose-optimizations are presented.

4 Reliable and Fast Distance Computation

The computation of distances between cylinders is fundamental to ensure safety margins for the optimization of MTCS. The evolutionary algorithm creates and evaluates several hundreds of constellations per second using the distance between cylinders as a core function for the detection of collisions. Therefore, the overall optimization process can be accelerated using faster methods for the distance computation. Furthermore, the number of tests can be reduced using adequate pretests and a smart implementation. The processing speed depends mostly on the number of the calls to complex functions and their run time. The computation of square roots, e. g., takes a long time compared to operations like multiplications or additions, which can be computed directly by the CPU. Thus, avoiding the extraction of roots by working with squared values wherever possible can speed up the computation dramatically. To reduce the number of function calls by using pretests, it is important to analyze application-specific requirements and adjust the use of the general algorithm.

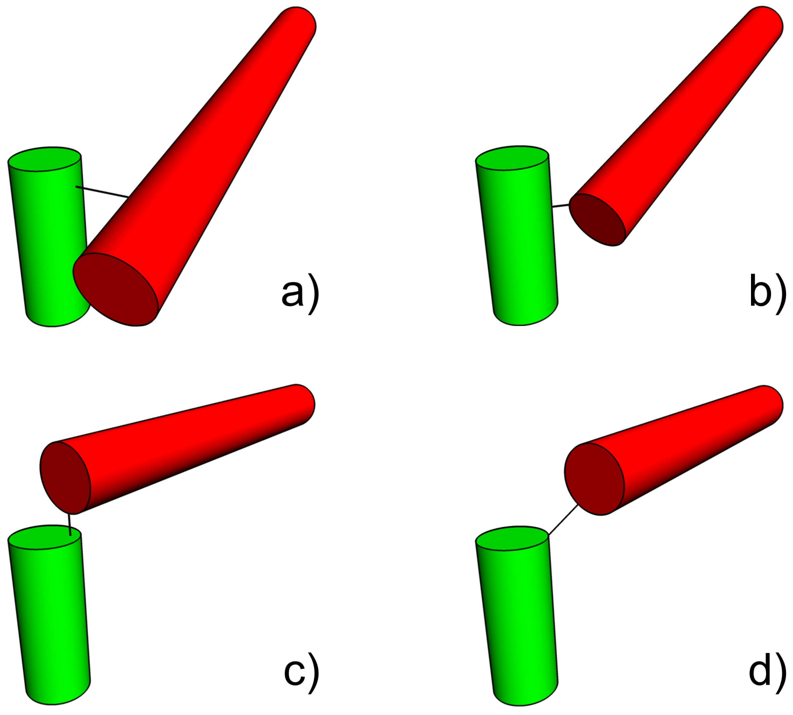

Figure 2: Four cylinder arrangements with the shortest distance between a) the shells, b) a shell and a circle border, c) a disk and a circle border, and d) two circle borders

As described before, the computation of the overall minimum distance between two cylinders can be realized by subdividing the cylinder surfaces into basic elements. Figure 2 illustrates the different cases for the distance computation. They can be distinguished by the locations of the points of closest proximity on the cylinder surfaces. In figure 2a, both shells are closest, whereas figure 2b demonstrates the situation where one shell and a circle border of the other cylinder surface are closest. The cases of figures 2c and 2d have minimal distance

Algorithm 1 Distance computation between two cylinders

```
procedure DistanceCylinders($CylA$, $CylB$, $d_{UpperBound}$)
    $radii \leftarrow CylA.Radius + CylB.Radius$
    ($d_{LS}$, $a$, $b$) ← DistanceLineSegments($CylA.LineSegment$, $CylB.LineSegment$)
    if $a, b \in ]0, 1[$ then
        return $d_{LS} - radii$                                ▷ min. distance between shells
    if $d_{LS} - radii \geq d_{UpperBound}$ then
        return $d_{LS} - radii$                                ▷ min. distance ≥ upper bound

    if any cylinder axis intersects with a disk of the other cylinder then
        return 0                                               ▷ interpenetration of cylinders

    $d_{UBDisk}[] \leftarrow$ UpperBoundDistancesBetweenDisks($CylA$, $CylB$)
    if any of $d_{UBDisk}[]$ = 0 then
        return 0                                               ▷ intersection of cylinder disks

    $d \leftarrow d_{UpperBound}$                              ▷ current minimal distance value
    $DClist \leftarrow \emptyset$                              ▷ sorted list of dist. computation objects
    for all pairs $\{C1, C2\}$ of circles do                   ▷ four pairs
        $d_{Disk} \leftarrow$ DistanceCircleDisk($C1$, $C2$)
        $d \leftarrow min(d, d_{Disk})$
        if $d_{Disk} = \infty$ then                            ▷ extremal circle points not above disk
            $d_{Disk} \leftarrow$ DistanceCircleDisk($C2$, $C1$)
            $d \leftarrow min(d, d_{Disk})$
            if $d_{Disk} = \infty$ then                        ▷ extremal circle points not above disk
                $d_{LBDisk} \leftarrow$ corresp. $d_{UBDisk}[] - radii$   ▷ calculate lower bound
                $DClist$.Insert($CircleCircleDC$($C1$, $C2$, $d_{LBDisk}$))
            end if
        end if
    end for

    for all pairs $\{Cyl, C\}$ of cylinders and circles do     ▷ four pairs
        $d_{LS} \leftarrow$ DistancePointLineSegment($C.Center$, $Cyl.LineSegment$)
        $DClist$.Insert($CircleCylinderDC$($Cyl$, $C$, $d_{LS} - radii$))
    end for

    while $DClist$ not $\emptyset$ do
        $DC \leftarrow DClist$.PopMinItem( )
        if $DC$.GetLowerBoundPretestDistance( ) $> d$ then
            return $d$                                         ▷ minimum distance found
        $d_{DC} \leftarrow DC$.ComputeDistance( )
        $d \leftarrow min(d, d_{DC})$
    end while

    return $d$
end procedure
```

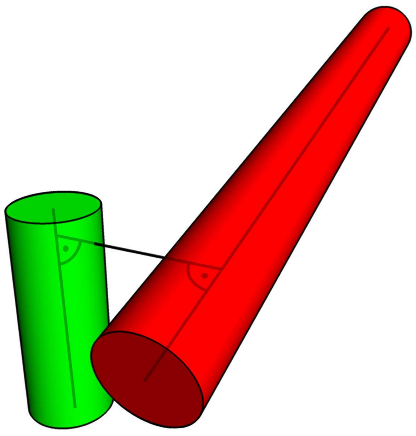

Figure 3: Distance of axes of cylinders for fast pretest

between two circle borders and between a circle border and a disk, respectively. Computing the minimum of the series of combinations of basic elements leads to the overall minimum distance between cylinder surfaces. Namely, one distance between the shells, eight different distances between circle borders and disks, four computations of circle-line distance and another four variants between circle and circle have to be considered and compared to get the distance between the cylinder surfaces.

Algorithm 1 depicts the implementation of the distance computation between cylinder surfaces, assuming the cylinders do not intersect. Therein, many procedures are used, delivering the minimum distance between basic geometric surfaces and curves. For the given preconditions, an initial distance computation considers only the line segments that are given by the axes of the cylinders. Therefore, the function DISTANCELINESEGMENTS is used, which computes the distance between two lines (the cylinders axes) with respect to the relevant part of the line that is defined by the length of the cylinders. Figure 3 visualizes the case where the points of minimum distance of the line segments are located inside their segment. Thus, the connecting passage is normal to both lines, and the minimum distance of the cylinder surfaces is given by the distance of the line segments reduced by both radii (see algorithm 1, line 5).

The result of the previous distance computation between line segments gives a lower bound $d_{LowerBound}$ for the distance between the surfaces of the cylinders when subtracting both radii of this value. Given an upper bound $d_{UpperBound}$ for distances that are relevant, the algorithm terminates if $d_{LowerBound} \geq d_{UpperBound}$ (cf. line 7). In this case, the algorithm performs very well with regard to speed and accuracy.

For the presented application, the precise distance might be less important since the realizability of two or more bores depend on the compliance of safety margins. Therefore, the described algorithm can often terminates at line 7. Since the computation of the distance between line segments performs very well, the overall computation is very fast in many cases, in contrast to an exact computation of the minimum distance.

Though the distance computation between a circle border and a disk as well as between

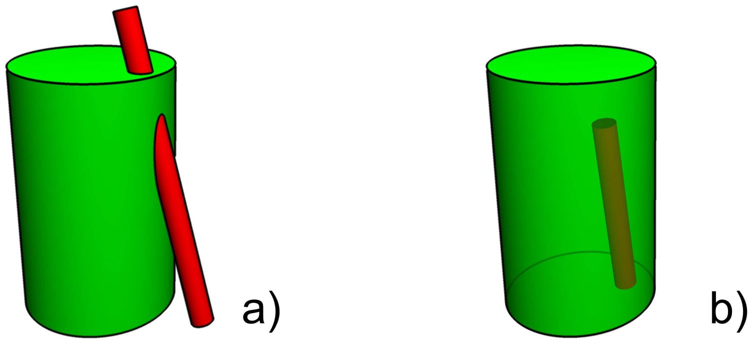

Figure 4: Degenerated cylinder arrangements with a) interpenetrating cylinders and b) a cylinder completely located inside the other

two disks is quite fast, pretests can be performed to decide which distance computations are candidates of giving the real minimum distance between cylinder surfaces. Additionally, tests have to be performed to ensure that disks of different cylinders do not intersect. Latter results from the consideration of only the cylinder surfaces though the cylinders should be regarded as solid. The pretest is implemented in the call to the procedure UPPERBOUNDDISTANCE-BETWEENDISKS, which returns zero if the disks intersect and otherwise the distance of the centers of the disks plus both radii. The algorithm may terminate in line 14 as well as in line 10 returning a distance of zero, if the cylinders intersect. Latter excludes the special constellation of cylinders, which is depicted in figure 4a where none of the described cases of figure 2 gives the real minimum distance.

If none of the previously described conditions hold, the minimum distance is either between two circles or between a circle and a shell. The first case can be subdivided further into the arrangements shown in figures 2c and 2d. For all pairs of circles of different cylinders, three computations have to be performed. The mutual minimum distances between the circle border and the disk of the other circle is computed (lines 19 and 22). The procedure DISTANCECIRCLEDISK returns an infinite if the case of figure 2c is not valid either because the closest point on the disk is not inside or if the connecting passage does not point the same direction as the outer normal vector of the circle. Otherwise, distance is computed by placing the disk centered in the x-y plane and determining the two points of the circle border with minimum and maximum z-value. Projecting both points onto the disk gives two distances between circle border and disk. The smaller one is the minimum distance between disk and circle border if the disks do not intersect. Otherwise, the distance is zero. The minimum and maximum point can be computed easily using the vector

$$\vec{v} = normal_{circle} \times normal_{circle} \times normal_{disk}.$$

Adding $\vec{v}$ or $-\vec{v}$ to the center of the circle gives the closest points between circle border and disk.

If neither of both distances between circle border and disk is valid, the minimum distance may be between both circle borders. The distance computation for two circle borders is computed by Vranek's numerical method. This distance computation is slow compared to previous computations (cf. algorithm 1 line 26, CIRCLECIRCLEDC). The same applies to the distance computations between circle and cylinder (line 33, CIRCLECYLINDERDC). Therefore, all the distance computations, which can lead to the overall minimum distance between cylinder surfaces, are inserted into a sorted list. The sorting criterion for this is the lower bound value for each case of distance computation. Thus, all inserted computations can be performed in ascending order since tests with a small lower bound might give smaller real distances. Furthermore, the current minimum distance can be compared to the lower bound of the next test (GETLOWERBOUNDPRETESTDISTANCE), indicating whether a diminishment of the minimum distance can be achieved with the remaining distance computations. The lower bound for the CIRCLECIRCLEDISTANCECOMPUTATION is given by the distance of the circle centers minus both radii, the one for the CIRCLECYLINDERDISTANCECOMPUTATION is determined by the distance of the center of the circle to the line minus both radii.

While performance tuning at the coding level can speed up the computation slightly, major enhancements can be achieved by avoiding tests as often as possible. The pretests and precomputations utilized provide good estimates what kind of results can be expected for the real distance computation. As the deviation from the precomputed distance is highly predictable it can be assumed that the smallest pretest distances in many cases lead to smallest real distances. Therefore, sorting the values and computing the distances with small pretest distances first can often reduce the amount of computations that have to be performed. Furthermore, the pretest distances are minimum bounds and, thus, the computation can stop if the next pretest value is bigger than the current minimum distance.

The evaluation of the current work was executed with several different test datasets. For each constellation, an iterative refinement procedure was used for the computation of the real distance between cylinders. The cylinders were approximated by line segments and for all pairs of segments the minimum distance was computed. With a hierarchical refinement and the exclusion of nonpromising regions, the accuracy for the computation of the distance could be increased until numerical limits are reached. This, however, is computationally expensive.

As described in section 3, an extensive test suite was build up. Different constellations from random and real scenarios were generated and modified randomly. Additionally, all test were benchmarked with different features of the algorithm enabled. Pretests, sorting, and upper bounds for the distance computation were combined. The results showed that the implemented features speed up the computation in almost any case with adequate accuracy. With real world data, most computation time can be saved using upper bounds.

5 Summary and Conclusion

The fast and accurate computation of minimal distances between cylinders is essential for a fast stochastic optimization of designs of mold temperature control systems. Though Vranek proposed the basic idea, a fast and accurate implementation for the distance computation between cylinders, a complete implementation covering every specific case was not available.

It has been shown, that a significant speedup of the original algorithm has been achieved by utilizing pretests and using a smart implementation. In addition, specific knowledge concerning the application at hand has led to further possibilities of reducing the average run times. Namely, an upper bound for the relevant distance was used.

In this paper a very fast and at the same time reliable and accurate algorithm has been developed. It completely covers every specific case and performs much faster than a direct implementation of Vranek's proposal. The reliability and accuracy were tested and ensured with a comprehensive test suite, which consists of three test scenarios: random tests, synthetic tests, and application driven tests. The random tests ensured the wide-range completeness of the algorithm, whereas synthetic test cases were used to assure critical constellations where numerical problems had been expected. The input data for the application at hand has been used in particular to confirm the high speed of the algorithm in practice.

Though the speed of the pure distance computation has already been improved significantly using pretests and advanced optimization techniques, the highest gain in speed has been achieved by avoiding as many tests as possible. A dramatic increase of speed has been realized by making use of the fact that the relevant range of distances is limited by an upper bound for the problem at hand.

The algorithm makes use of algebraic and numerical functions to determine the distances. The accuracy of the latter functions was parameterized giving a scalable algorithm. This allows an adjustment to different application scenarios since the run time of the costly numerical computations increases with the accuracy.

All in all, the algorithm has shown to be very fast, accurate, and reliable and can be adopted to different application requirements.

References

[1] Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[2] David Vranek. Journal of graphics tools – papers – fast and accurate circle-circle and circle-line 3d distance computation. http://jgt.akpeters.com/papers/Vranek02/, February 2009.

[3] Klaus Weinert and Jörn Mehnen and Thomas Michelitsch and Karlheinz Schmitt and Thomas Bartz-Beielstein. A multiobjective approach to optimize temperature control systems of molding tools. *Production Engineering Research and Development, Annals of the German Academic Society for Production Engineering*, XI(1):77–80, 2004.

[4] Christian Lennerz and Elmar Schomer. Efficient distance computation for quadratic curves and surfaces. In *2nd IEEE Conf: on Geometric Modelling and Processing*, pages 60–69, 2002.

[5] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. *Numerical Recipes in C.* Cambridge University Press, Cambridge, UK, 2nd edition, 1992.

[6] Philip J. Schneider and David H. Eberly. *Geometric Tools for Computer Graphics*. Elsevier Science, San Francisco, 2003.

[7] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. Wiley-Interscience, NY, USA, 1995.

[8] David Vranek. Fast and accurate circle-circle and circle-line 3d distance computation. *Journal of Graphics Tools*, 7(1):23–32, 2002.